NAVAL POSTGRADUATE SCHOOL
Monterey, California

THESIS

SOFTWARE SYSTEM REQUIREMENTS FOR THE ARMY TACTICAL MISSILE SYSTEM (ATACMS) END-TO-END SYSTEM USING THE COMPUTER AIDED PROTOTYPING SYSTEM(CAPS) MULTI-FILE APPROACH

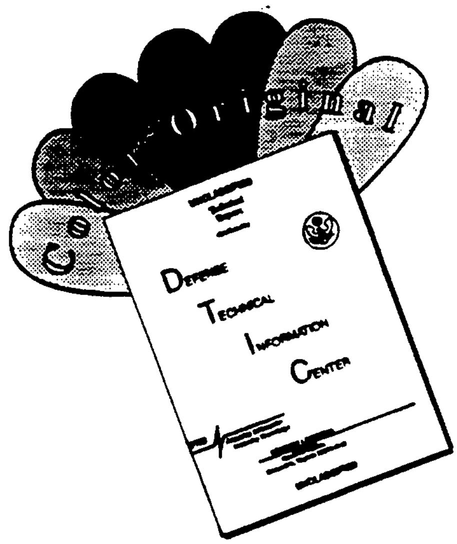
by

David Stuart Angrisani
George Steven Whitbeck

September 1996

Thesis Advisors: Luqi
Valdis Berzins
Man-Tak Shing

DISCLAIMER NOTICE



THIS DOCUMENT IS BEST QUALITY AVAILABLE. THE COPY FURNISHED TO DTIC CONTAINED A SIGNIFICANT NUMBER OF COLOR PAGES WHICH DO NOT REPRODUCE LEGIBLY ON BLACK AND WHITE MICROFICHE.

REPORT DOCUMENTATION PAGE

Form Approved OMB No. 0704-0188

1. AGENCY USE ONLY *(Leave blank)*	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	September 1996	Master's Thesis

4. TITLE AND SUBTITLE
Software System Requirements for the Army Tactical Missile System (ATACMS) End-to-End System Using the Computer Aided Prototype System (CAPS) Multi-File Approach

5. FUNDING NUMBERS

6. AUTHOR(S)
David Stuart Angrisani, George Steven Whitbeck

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Naval Postgraduate School
Monterey, CA 93943-5000

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)

10. SPONSORING / MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES
The views expressed in this thesis are those of the authors and do not reflect the official policy or position of the Department of Defense or the U.S. Government.

12a. DISTRIBUTION / AVAILABILITY STATEMENT
Approved for public release; distribution unlimited.

12b. DISTRIBUTION CODE

13. ABSTRACT *(maximum 200 words)*

The Department of Defense (DOD) is seeking software system requirements for the Army Tactical Missile System (ATACMS) End-to-End System, which comprises both ATACMS and all sensors, links, and command centers which enable integration across system and service boundaries. The complexity, multiple interfaces, and joint nature of planned ATACMS operations demands accurate specification of software system requirements. DOD also desires automated tools capable of developing rapid prototypes to assist in system definition and reduce system risk.

The goals of this thesis are to provide a rigorous model which can be utilized to validate current specifications, and, to demonstrate CAPS on a large scale project. Accomplishment of these two would provide a needed corroboration of the ATACMS specification, as well as move CAPS out of the purely academic environment.

The result of this thesis is mixed. Due to a paucity of data from which to derive the requirements, the model is generic in nature and is in need of significant customer evaluation, which is not forthcoming. However, CAPS demonstrated its fundamental concept within the bounds of the project, with refinements in code generation, interface, and graphics either incorporated or identified. CAPS is ready for use on an actual project by an experience team of systems analysts.

14. SUBJECT TERMS
CAPS, PSDL, Systems Analysis, Software Requirements, Rapid Prototyping, TAE Multi-file

15. NUMBER OF PAGES
171

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	Unclassified	Unclassified	UL

NSN 7540-01-280-5500

Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std. 239-18
298-102

Approved for public release; distribution is unlimited

SOFTWARE SYSTEM REQUIREMENTS FOR THE ARMY TACTICAL MISSILE SYSTEM (ATACMS) END-TO-END SYSTEM USING THE COMPUTER AIDED PROTOTYPING SYSTEM (CAPS) MULTI-FILE APPROACH

David Stuart Angrisani
Lieutenant Commander, United States Navy
BS Nathaniel Hawthorne College, 1982

George Steven Whitbeck
Major, United States Marine Corps
BS United States Naval Academy, 1983

Submitted in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE IN COMPUTER SCIENCE

from the

NAVAL POSTGRADUATE SCHOOL

September 1996

Authors:

David S. Angrisani

George S. Whitbeck

Approved by:

Luqi, Thesis Advisor

Valdis Berzins, Thesis Advisor

Man-Tak Shing, Thesis Advisor

Ted Lewis, Chairman, Department of Computer Science

ABSTRACT

The Department of Defense (DOD) is seeking software system requirements for the Army Tactical Missile System (ATACMS) End-to-End System, which comprises both ATACMS and all sensors, links, and command centers which enable integration across system and service boundaries. The complexity, multiple interfaces, and joint nature of planned ATACMS operations demands accurate specification of software system requirements. DOD also desires automated tools capable of developing rapid prototypes to assist in system definition and reduce system risk.

The goals of this thesis are to provide a rigorous model which can be utilized to validate current specifications, and, to demonstrate CAPS on a large scale project. Accomplishment of these two would provide a needed corroboration of the ATACMS specification, as well as move CAPS out of the purely academic environment.

The result of this thesis is mixed. Due to a paucity of data from which to derive the requirements, the model is generic in nature and is in need of significant customer evaluation, which is not forthcoming. However, CAPS demonstrated its fundamental concept within the bounds of the project, with refinements in code generation, interface, and graphics either incorporated or identified. CAPS is ready for use on an actual project by an experienced team of systems analysts.

TABLE OF CONTENTS

ACKNOWLEDGMENT

To Dr. Luqi, Dr. Berzins, and Dr. Shing, we would like to express our most profound thanks for the many hours of patient guidance. We stand in professional admiration and are most grateful for our association with the CAPS group. We wish you all the best in your future endeavors.

To our wives, Tammy and Cathy, and our children, Matthew, Christine, Jenny, and Michelle, we owe you a debt that can never be repaid. To you, who have always provided the loving homes where we may seek refuge and fulfillment, go our fullest measure of love and thanks.

To a loving and gracious God, we give humble thanks and praise, knowing that without Your blessing and guidance none of this would matter or be possible.

I. INTRODUCTION

A. PROBLEM STATEMENT

The requirements of modern joint warfare have driven the Department of Defense (DOD) to seek improved interoperability between systems of the different services as well as more responsive and flexible employment of intra-service systems. Additionally, the advent of modern technological warfare has placed a new emphasis on Command, Control, Communications, Computers, and Intelligence (C4I). Combat users require the ability to access and utilize data from a variety of platforms in an environment of ever more stringent time and accuracy demands. The ability to deliver real-time information from sensor to shooter is the primary goal of the new integrated systems which will be fielded in the next century.

Coexistent with the increased operational capabilities is an increase in the complexity and difficulty of defining, designing, and developing complex weapon systems. DOD recognizes the current inadequacy of their requirements generation process. DOD desires the evaluation of automated tools to assist the systems analyst in rapidly developing accurate systems requirements.

The topic of this thesis is the software system requirements needed for the development and deployment of the new, long range Army Tactical Missile System (ATACMS) and the other C4I systems which together comprise what is termed the "End-to-End" System. The new, longer ranged ATACMS necessitates the incorporation of other services' systems in order to fully achieve its maximum combat capability but this need also makes systems analysis more difficult.

The placement of the ATACMS within a "system of systems" brings with it several programmatic difficulties. Firstly, without any single cognizant technical authority to be responsible for the entire system, each system component is developed and deployed without planned integration with any of the others. Secondly, testing, either developmental or operational, is difficult and often prohibitively expensive. Thirdly, systems are integrated after their requirements are fixed, rather than defining generic requirements to which new components can be designed. Fourthly, the scale and complexity of these systems require the use of computer based tools to assist the analyst in accomplishing the analysis.

To address these difficulties on the upgraded ATACMS project, the Director, Test, Systems Engineering & Evaluation developed a Memorandum of Agreement (MOA) [Ref. 1] which attempts to encompass the entire End-to-End System and bring together all parties responsible for fielding a successful long range ATACMS with interoperability capabilities. The purpose of this MOA is to focus on interface issues. In an attempt to minimize or eliminate complete system testing, the DTSE&E is interested in modeling and simulations which could assist in determining and verifying interface and component requirements.

To this end, the Computer Aided Prototyping System (CAPS) research group at the Naval Postgraduate School (NPS) is assisting in evaluating and refining the software system requirements for the ATACMS End-to-End System as well as demonstrate the capabilities and suitability of CAPS on a large real world system.

This thesis analyzes the ATACMS system, specifies system requirements, identifies unknowns and constructs a model using CAPS which demonstrates those requirements in a operating model so that the requirements can be verified and refined through subsequent iterations. Additionally, CAPS is evaluated as a requirements generation tool.

B. RAPID PROTOTYPING & CAPS

The use of prototyping in engineering hardware has a long and successful legacy but is relatively unused in software development. The application of software prototyping to the ATACMS requirements analysis enables us to ascertain the vital system attributes without completely specifying or writing the code for the entire system. In fact, prototyping is most appropriate on systems lacking strong definition. Since the target system is not that well defined it becomes necessary to make several iterations of the model, each being verified by the users as to correctness and suitability.

This use of a prototype model constructed rapidly and efficiently is only possible through the use of computer assistance. The CAPS system provides a set of integrated tools optimized for the rapid development of reliable and accurate real-time prototypes. These tools allow the developer to design, construct, execute and debug the prototype.

C. METHODOLOGY & DELIVERABLES

The general methodology used to develop the requirements and construct the prototype consists of analyzing pertinent documents defining and describing the End-to-End System components to determine the essential system attributes and constraints on the architectural level. From this analysis a series of models based on a single instance of a "sensor to shooter" path is constructed, which identifies known system requirements, identifies system significant unknowns, and where necessary, incorporates substitute requirements to keep the thesis at the UNCLASSIFIED level. Wherever possible, generic requirements are developed so as not to limit the model's utility to a single instance. The essence of the work is in its simplicity and ability to distill only the essentials into the model.

The deliverables are an executable prototypes (with an non-executing extension), the prototyping language description for each, source code, and this thesis write-up which includes an evaluation of CAPS as a requirements generation tool. The models are available for review by any of the DOD and ATACMS stakeholders, with the intention of performing subsequent refinements as well as the substitution of actual classified parameters.

D. ORGANIZATION OF THESIS

In addition to this introduction the thesis contains Chapter II which provides sufficient background information to make the thesis a stand-alone document. Chapter III describes the analysis and the three iterations of the models in detail. Finally, Chapter IV provides the conclusions of the research and some recommendations for follow-on work.

II. BACKGROUND

A. TASKING

In 1994, the Office of the Under Secretary of Defense began development of a Memorandum Of Agreement (MOA) which would establish, among other items, the system level Critical Operational Issues (COI) involved in the operation of the Army Tactical Missile System (ATACMS) End-to-End System. [Ref. 1] The End-to-End System contains those constituent components which allow the ATACMS to access, use, and be used by other units (or even other services) than the one to which it is assigned. This concept, known as "sensor to shooter", consists of a high degree of interoperability and connectivity, allows each commander a wider range of options and each user access to more applicable and timely data, and acts as a force multiplier.

The MOA was necessary since many of the assets involved in successful employment of the ATACMS are outside the cognizance of the Army. Recognizing that the majority of problems do not occur in the technical performance parameters of individual systems, but instead involve interoperability and connectivity issues [Ref. 2], the Office of the Director, Test, Systems Engineering & Evaluation (ODTSE&E), approached the CAPS research group at NPS about using CAPS to assist in defining requirements and identifying COI's for the ATACMS End-To-End System while at the same time demonstrating CAPS on a large real world problem. An initial request to use CAPS as a tool for Developmental Test & Evaluation was also made, but later deleted from the tasking.

The Draft MOA defined the ATACMS End-to-End System from the battlefield architecture for deep operations as defined in the U.S. Army's Integrated Battlefield Targeting Architecture (IBTA) Handbook. For the purposes of defining the scope of the evaluation, the ATACMS End-to-End System was defined as follows:

- **Target Acquisition (TA)** systems include the Special Operations Forces (SOF), GUARDRAIL Common Sensor (GRCS), Joint Surveillance Target Attack Radar System (JSTARS), Unmanned Aerial

Vehicle (UAV), and other theater, Army, and national surveillance assets. Target acquisition systems include the procedures, people, communications and automated systems in the Analysis and Control Element at Corps level.

- **Fire Support Command, Control, and Communication (FSC3)** systems include the procedures, people, communications and automated systems in the Deep Operations Coordination Cell (DOCC) and echelons below Corps that provide targeting and fire control information to the firing batteries.

- **Fire Mission Execution (FME)** systems include the procedures, people, communications and equipment at the MLRS battery that execute fire missions and the performance of the missiles and submunitions. [Ref. 3]

ODTSE&E has a particular concern with the interface between the TA and the FSC3 elements since those are the least defined with respect to scenarios, testing criteria, and requirements research. [Ref. 1]

B. JUSTIFICATION

Models of large multi-platform systems can be correctly represented by several methods whose approach is radically different. Each method has its strengths and weaknesses. While appropriate and adequate for defining the scope of the work, the delineations contained in the MOA were not a suitable basis for the models. The general approach is to define the system according to its modular functionality, thus allowing the greatest latitude for future modification and also making interpretation and comment by the users easier. The modeling process was begun with the intent to model the CAPS operators using generic descriptions which enables the system in a very simple manner to represent most, or possibly all, the possible scenarios. Instead, the focus on a single instance of "sensor to shooter" is used since the attributes of each individual object in the system are very similar to those of other objects performing the like function (further refinement into a generic model is anticipated in future work).

A second, and perhaps more important and constraining difficulty, is the lack of a unified point of contact through which to access the information needed to ensure that the represented systems are being accurately modeled. The tasking office (ODTSE&E) has no demonstrated technical background and appears to be a reporting point for the various programs involved in fielding the End-to-End System. Additionally, much of the information needed is highly classified, unattainable or in aggregate beyond the scope of acquisition from NPS. Based on our experience with fielded military systems and with our modeling experience in the CAPS group, a decision was made to make a best estimate of the requirements for the End-to-End System, providing our own specific parameters and abstractions. Where and how these were used will be noted. To make the model truly useable the parameters of the actual components will have to be supplied, verified, or modified by the user(s).

A third difficulty encountered is the use of a single platform for the CAPS system. The ATACMS End-to-End System is a multi-platform integration of distinct operational subsystems. It is possible, within limits, to abstract the operation of each component so as to represent its essential characteristics without having to perform each individual function of that component, and thus to some extent overcome this multi-platform/single processor dichotomy. Due to the low quantity of operators, we have not been particularly constrained in our modeling using this method. However, as the operator count (and subsequent detail) grows, at some point it is likely that our operator loading, combined with the demands of the UNIX system will exceed our single processor capability.

Given the above, the model which is submitted is a single instance of the ATACMS End-to-End System using parameters supplied in part by the documentation and in part by the researchers. The purpose of this model is to demonstrate that a multilevel model could be quickly constructed and to pass data successfully through this model, while characterizing/identifying the most important attributes of the operators. In essence, this is the skeleton on which the details of subsequent iterations will be hung.

C. GOALS

Broadly defined, the goal of the research is to meet the tasking as delineated while demonstrating CAPS. The Critical Operational Issues for the three functional areas described above (IIA) and which would impact model performance are as follows:

- **Target Acquisition (TA)** Do target acquisition systems provide adequate, timely, and sufficient target location data to FSC3 systems to effectively employ the ATACMS variants?

- **Fire Support Command, Control, and Communication (FSC3)** Do FSC3 systems provide timely and sufficient mission execution instructions to effectively employ the ATACMS variants?

- **Fire Mission Execution (FME)** Do the ATACMS variants achieve the levels of lethality and effectiveness specified by the appropriate requirements documents? [Ref. 3]

Obviously, these are very general questions from which to ascertain specific and verifiable goals. Though included in the project Goals Hierarchy for completeness, the COI for FME was disregarded in the model construction. These COI are mainly concerned with hardware and physical constraints that are well defined. There are several high fidelity computer models already in service to use as a testing base for this element. [Ref. 4]

The goal of the research is to perform a requirements analysis using CAPS which would further define, and, where possible address, the COI's for above and provide the following specifics [Ref. 5]:

- A simplified model of the system's environment.
- A description of the system goals hierarchy and the functions it must perform.
- Performance constraints on the system.
- Implementation constraints on the system.
- Resource constraints for the development project.
- The specification of the external interfaces of the major components.

D. INTEGRATED BATTLEFIELD TECHNOLOGY ARCHITECTURE (IBTA) HANDBOOK

The primary source for information about the ATACMS End-to-End System components comes from a U.S. Army publication series called the Integrated Battlefield Technology Architecture Handbook (IBTA) [Ref. 6]. Work on the IBTA was begun in 1992 at the direction of US Army Assistant Deputy Chief of Staff (Operations and Force Development). The purpose of the IBTA is to articulate an integrated battlefield targeting architecture from the command post view which addresses brigade↔corps horizontal and vertical integration requirements for maneuver, fire support, Intelligence and Electronic Warfare (IEW), air defense, and command post operations. The resulting "golden threads" establish the requirements, interfaces, and throughputs to allow data exchange at the speed and accuracy required.

The IBTA delineates current and planned C4I systems either employed by or integrated with the Army systems. Force architectures for the years 1994, 1999, and 2010 are included to show the progression from currently fielded systems through those in the acquisition pipeline, and then out to those systems in the definition and concept exploration phase.

The IBTA shows the architecture relationships between units/systems from the national level all the way down through the individual unit. The main publication in the series is classified SECRET. Its UNCLASSIFIED topical contents are (actual parameters are classified):

- An overview of each subsystem, including current capabilities, projected future enhancements and a classified assessment of the system.
- Issue specific assessments synopsizing targeting issues by proponent in fact sheet format.
- Sensor-to-Shooter "golden thread" vignettes identifying targeting data paths for selected systems.

The vignettes provide a text description of each "golden thread" and analysis of the data path in five levels of detail:

- Level I- Macro architecture.
- Level II- A listing of components in the data path.
- Level III- Approximate timings for operators, processors, and transmissions.
- Level IV- Protocol, baud rates, formats, and frequency spectrums.
- Level V- Additional "next-node" transmission data.

The other primary product in the series is the Architecture Annex which shows in graphical form the architecture relationships between all systems from Corps to Division/Brigade, as well as the individual Corps, Division, and Brigade Architectures. The architectures are also presented by functional areas such as artillery, air defense, and intelligence. The artillery architectures are particularly useful for analyzing the ATACMS End-to-End System since the inapplicable architectures are absent. [Ref. 6] [Ref. 7]

E. ATACMS END-TO-END SYSTEM

The ability to use the Army TACMS to support deep operations has prompted the Army to begin viewing the ATACMS more broadly, as one component of an integrated system which can provide "sensor to shooter" capability from a wide assortment of reconnaissance and sensor platforms through a decentralized control system to widely separate ATACMS batteries. This employment enables more flexible use and response by field commanders.

In this section, the various components which can and will be expected to integrate into the End-to-End System are discussed. In general the system has been divided into four functional areas: sensors, command center, links, and shooter. Detailed description of the actual ATACMS firing element architecture (as opposed to the End-to-End System) is not included, as mentioned earlier, due to its system maturity. The links are between sensor-command center, command center-shooter, or are contained within the described subsystems (important for decomposition).

1. Sensors

The IBTA contains all current and planned sensors which associated field units will encounter. Some of these sensors are used primarily for purposes other than artillery strikes and are not included in this discussion. The range of sensors include ELINT, SIGINT, and HUMINT sources. Some of the assets are Army, some are Air Force and some are national assets. Detailed descriptions of selected sensors are detailed below within the limits of classification. [Ref. 6] [Ref. 7] Refer to Figure (1).

- **Joint Surveillance and Target Attack Radar System (JSTARS)**
  This Air Force airborne sensor provides continuously updated data on enemy force moving vehicles. Data provided includes direction, location, numbers and rates of movement of enemy vehicles. It also senses vehicular traffic associated with command posts and air defense sites. No target ID or correlation is provided. A Moving Target Indicator (MTI) is provided. Data is collected across a corps area using a limited Synthetic Aperture Radar (SAR). The current JSTARS interfaces with the Ground Station Module (GSM), however future versions will accommodate the Common Ground Station (CGS) as well as being augmented by the Joint Tactical Information Distribution System (JTIDS). This system is particularly applicable to deep ATACMS operations.

- **Trailblazer/Teammate**
  This is a division level communications intelligence (COMINT) sensor which provides radio location voice reports to the division Analysis and Control Element (ACE). This system is being phased out in favor of the Ground Based Common Sensor (GBCS) described below.

- **Ground Based Common Sensor (GBCS)**
  This sensor provides an all weather, day/night, on-the-move, automated and integrated COMINT, ELINT, EW suite for HF, VHF, UHF, SHF, EHF frequency bands. The GBCS is 100 percent interoperable with the Army QUICKFIX (AQF), US Marine Corps Mobile Electronic Warfare Support System (MEWSS), US Navy fast attack submarines, and selected Navy combatants.

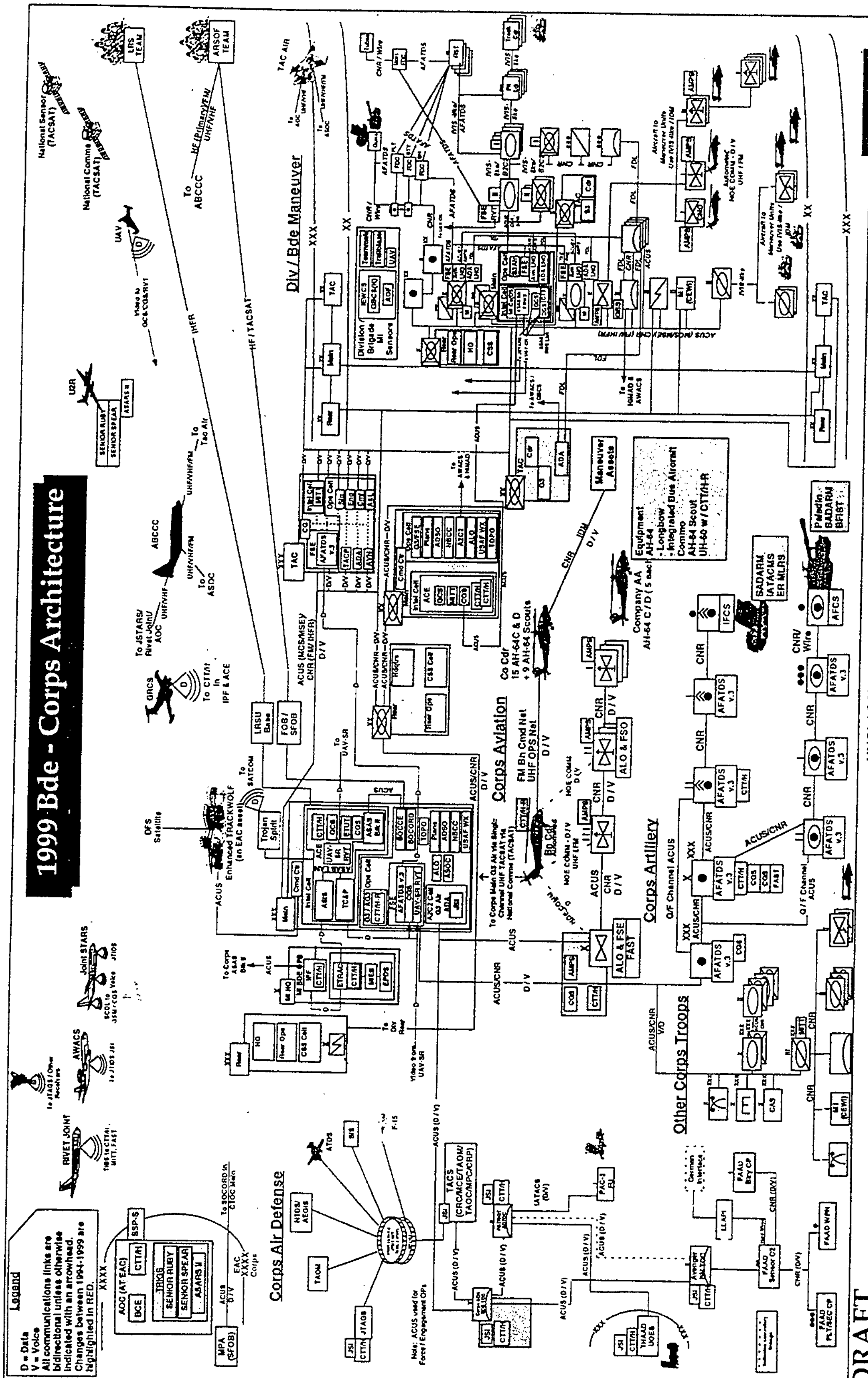


Figure 1. 1999 Brigade to Corps Architecture [Ref. 7]

- **Trackwolf/Enhanced Trackwolf**
  This sensor provides COMINT and DF against threat HF emitters to Echelons Above Corps (EAC) and national security agencies. This intelligence supports analysis of deep situation and target development. Target information is passed via fiber digital data networks. The sensor is particularly applicable to deep ATACMS operations.

- **Guardrail Common Sensor (GRCS)**
  This sensor is a corps level airborne SIGINT system capable of acquiring both communications and non-communications emitters. It digitally interfaces with the Commander's Tactical terminal (CTT).

- **Unmanned Aerial Vehicle (UAV)**
  This sensor provides real-time imagery of the battlefield via the CGS which supports corps and lower echelons with targeting, target damage assessment, and battlefield management information. It interfaces with the CGS.

- **Long Range Surveillance Teams (LRS)/Special Operations Forces Teams (SOF)**
  These provide a corps with highly reliable HUMINT collection on enemy activities from concealed observation posts employed against second echelon and follow-on forces. Information is generally consolidated and transmitted at predetermined times via burst transmissions on a secure Improved High Frequency Radio (IHFR). LRS/SOF data would be particularly applicable to deep ATACMS operations.

2. Command Centers

The IBTA contains all current and planned command centers/command posts/ headquarters from Echelons Above Corps (EAC) down through the individual battery commands. The command centers are very much aggregate entities in as much as the "modular" equipment contained in a particular command center may be found at many different levels of the command structure thus allowing several elements in the architecture to perform similar/redundant functions. In some cases the command centers can be customized to meet a particular need.

This section contains those C4I elements involved in receiving, processing and disseminating targeting information and fire missions. Some communications links, while physically part of the command centers, will be treated separately in the next section on links. Detailed descriptions of selected elements are detailed below within the limits of classification. Refer to Figure (1). [Ref. 6] [Ref. 7]

- **All Source Analysis System (ASAS)** This is a multi-source processing facility located at the corps level. It provides automated assistance (fully automated in Block II) in analyzing, processing, and disseminating information and commands to other units. The system is capable of displaying and monitoring enemy locations, movements, and identification tags. The system provides for multi-sensor cueing, correlation, and fusion of data. The ASAS is part of the Analysis and Control Element (ACE) located the Intelligence Cell at corps headquarters.

- **Ground Station Module (GSM)**
This is the processing station for the JSTARS test platform and has had limited deployment during developmental testing. This station is currently located at corps and division level headquarters.

- **Common Ground Station (CGS)**
This is a multi-source processing station which is the follow-on to the GSM. It will be capable of intelligence analysis of JSTARS NTI and other imagery as available. The station will include a CTT for multi-platform access, correlation, and fusion in near real-time (NRT).

- **Commander's Tactical Terminal (CTT)**
The unit provides target quality data in NRT at selected critical nodes. Provides simultaneous full duplex data and half duplex voice communications between selected theater sensors and deployed CTT receivers. The CTT can transmit in individual, group, or broadcast modes. The CTT can receive the Tactical Information Broadcast System (TIBS), Tactical Related Applications (TRAP), Tactical Reconnaissance Intelligence Exchange System (TRIX), Unmanned Aerial Vehicle (UAV) imagery, and ASAS output.

- **Advanced Field Artillery Data System (AFATDS)**
An enhancement to the Tactical Fire Direction System (TACFIRE) which provides increased automation for command and control of

indirect fires. It accommodates all existing combat net radios (CNRs), LANs, and fielded data communications systems and equipment. It provides seamless vertical and horizontal interface for command and control of fires.

3. Communications Links

The IBTA contains all current and planned communications links used to connect the various architectures. The links include voice and data as well as RF, land-line, and LANs. Detailed descriptions of selected elements are detailed below within the limits of classification. Refer to Figure (1). [Ref. 6] [Ref. 7]

- **Joint Tactical Information Distribution System (JTIDS)**
  This is an advanced line of sight (LOS) radio for intra-theater tactical information distribution. Transmission occurs at prearranged times, with the radio receive capable during other periods. One terminal in the JTIDS network acts as a time reference. The system is very expensive and is currently only planned for deployment with the air defense elements.

- **Tactical Information Broadcast System (TIBS)**
  This system provides NRT information via an area broadcast. Producers on TIBS provide sanitized, fused information to passive receivers. This information is presented on a user-filtered, graphics oriented display. The CTT is capable of receiving and displaying TIBS information.

- **Surveillance & Control Data Link (SCDL)**
  A unique data link system used by the JSTARS test bed. In the production version this is to be augmented by JTIDS. The GSM, and later the CGS will be capable of receiving SCDL information.

- **Combat Net Radio (CNR)**
  A series of radios with varying capabilities which when combined form a loose network of RF communications. The Single Channel Ground-Air Radio System (SINCGARS) is a secure frequency hopping LOS radio capable of voice and data transmissions. The Single Channel SATCOM (S/C SATCOM) is a longer ranged CNR employed by remote users.

- **Common Data Link (CDL)**
  A family of modular communications hardware which provides secure data link for SIGINT and IMINT from/to all linked elements. The Army uses the CDL for communications between the GRCS and the the Integrated Processing Facility (IPF), and also the U2R airborne ASARS and the Enhanced Tactical Radar Correlator (ETRAC).

4. ATACMS Firing Element

The IBTA contains all architectural levels between corps and the actual firing battery. Though there are many different configurations possible, all are predicated on use of the AFATDS. Since this system is already fielded and has received wide operational use, the architecture below the AFATDS interface is not critical to understanding and quantifying the attributes and behaviors of the End-to-End System. Refer to Figure (1). [Ref. 6] [Ref. 7]

F. REAL-TIME SYSTEMS

When viewed as a total system, ATACMS, as described in the previous section, encompasses behaviors which are commonly referred to as real-time. There are many different definitions of real-time systems. The following is a good general description:

> "In real-time computing the correctness of the system depends not only on the logical result of the computation but also on the time at which the results are produced" [Ref. 8]

If these timing constraints are not met, then a failure has occurred. Hence, it is essential that the timing constraints of the system are strictly observed, which in turn requires that the system be predictable and reliable. Real-time systems may be further categorized as follows:

> A system where "performance is degraded but not destroyed by failure to meet a response time constraint is referred to as a soft real-time system." Systems where "failure to meet a response time constraint leads to system failure is a hard real-time system" [Ref. 9]).

The definitions of "hard" and "soft" are not universally defined. An alternate definition uses "hard" and "soft" to describe the degree of time constraint. Also, real-time systems are not "perfect" or "bug free" systems, but rather have well defined failure rates and resultant behaviors. In fact, assessment of failures with respect to system definition can be one of the major goals of prototyping.

One of the fundamental properties of real-time systems is "that some or all of its input arrives from the outside world asynchronously with respect to any work that the program is already doing" [Ref. 10]. The program must be able to block its current activity and then execute some other task, and when done, it must return gracefully to the previous task . Executing several tasks in what is, or may appear to be, in parallel (parallel vs multitasking), is a key characteristic of all real-time systems.

It is important to emphasize that "real-time" is not synonomous with "fast". It is not the latency of the response that is at issue (it could be of any magnitude), but "the fact that a bounded latency sufficient to solve the problem at hand is guaranteed by the system. In particular, it's frequently true that algorithms that guarantee bounded latency responses are less efficient," and thus slower, than algorithms that do not. [Ref. 11]

During the ATACMS analysis we identify the real-time attributes in the system as well define component and system failure rates and modes.

G. COMPUTER AIDED PROTOTYPING SYSTEM (CAPS)

In the classic approach to developing software, known as the waterfall method, a development project proceeds in discrete phases each of which consists of analysis, design, implementation, and testing. An essential characteristic of this approach is the thorough definition of requirements prior to the commencement of implementation. Any problems encountered in the testing phase are discovered only after significant investment in time and money. [Ref. 12]

Rapid prototyping is an alternative method which allows the quick development of an executable pilot program which can then be reviewed by the user for accuracy.

Through repeated iterations of the prototyping cycle (Figure 2) the user validates the requirements of the proposed system. Upon validation, the requirements serve as a basis for production software. In many cases, the prototype serves as a starting point for production code. However, it is important to remember that there are some important distinctions between the prototype and the production version [Ref. 13]:

- The prototype may not include all aspects of the production version.
- Prototype resources may not be available in actual operating environment.
- The prototype may have limited capacity.
- The prototype might meet timing constraints only with respect to linearly scaled time.

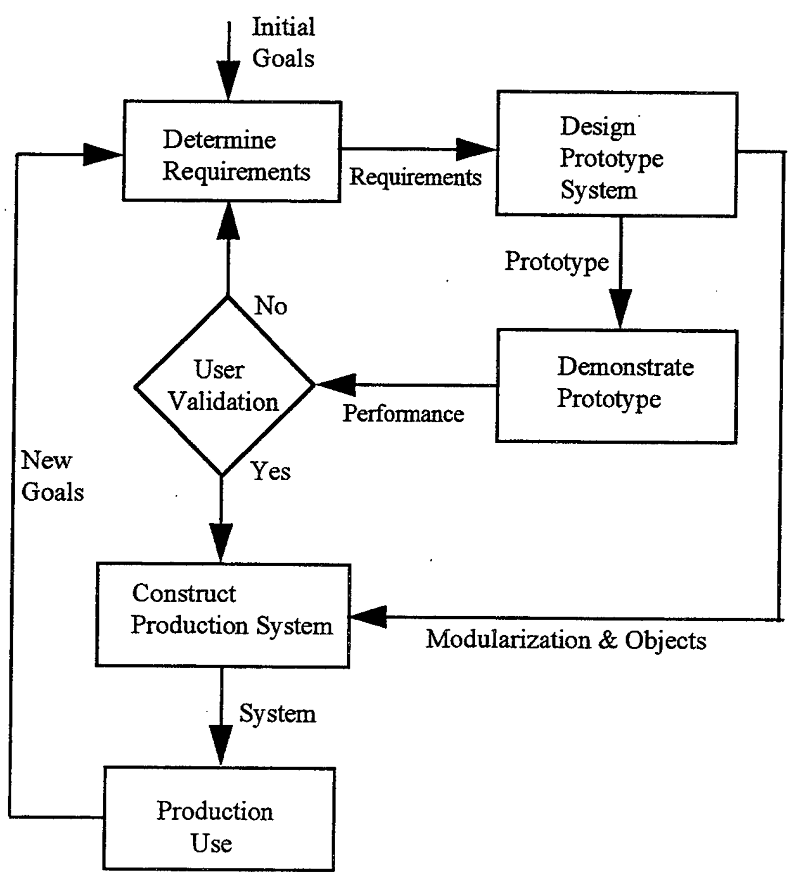


Figure 2. The Prototyping Cycle [Ref. 13]

The Computer Aided Prototyping System (CAPS) is an integrated software development environment which is at the heart of the requirements generation process used in this thesis. The Prototype System Description Language (PSDL) in CAPS is designed for specifying hard real-time systems. The PSDL descriptions produced by CAPS provide a formal and unambiguous definition of the modeled system.

CAPS consists of four major components: a set of editors for design entry, a software base of reusable components, and an execution support system to build the executable prototype (Figure 3).

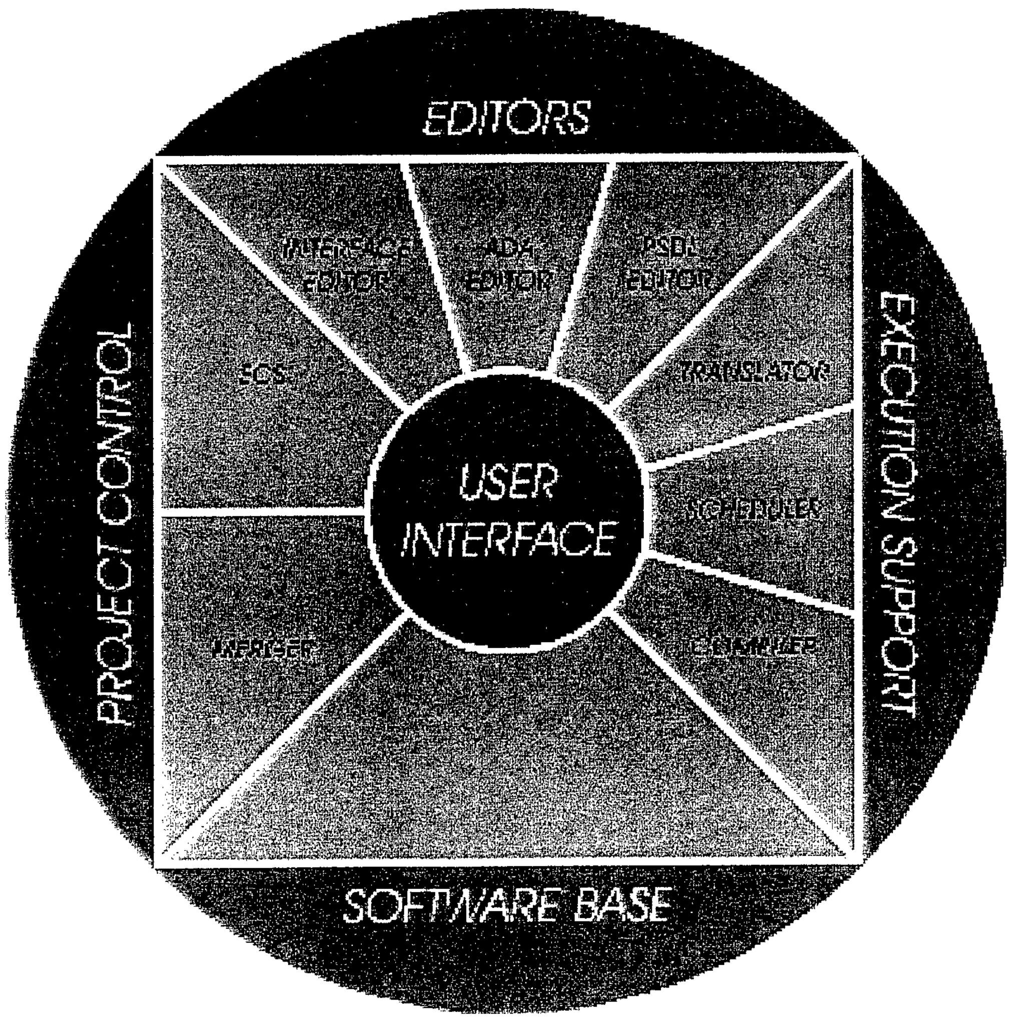


Figure 3. The CAPS System [Ref. 14]

By using the CAPS graphics and text editors the user can create a prototype which specifies the essential requirements of the system. The editor enforces consistency and enables rapid construction of a rigorous and accurate model. The system uses the Transportable Application Environment (TAE) to build a medium fidelity graphical user interface (GUI) for the prototype. Once the prototype is specified the user translates and schedules the prototype and then automatically creates an executable driver program which incorporates the requirements of the specification file. [Ref. 15]

CAPS also contains a software database system which consists of a software database, design database, a software reuse facility, automated system management, and version control. The software database tracks PSDL descriptions and ADA implementations for all CAPS reusable software components. The design database allows management coordination of concurrent team design efforts.

One of the most important issues to the prototype designer using CAPS is the treatment of real-time constraints identified during the analysis. It is important to understand CAPS' behavior in this respect. Atomic operators in a CAPS data flow diagram become ADA procedures in the implementation. Prioritizing these procedures into time critical (high priority) and non-time critical (low priority) is fundamental to specifying a real-time prototype. Determination of time criticality is an integral part of the systems analysis. In CAPS, criticality is represented by the assignment of a Maximum Execution Time (MET). The MET is the longest period between the time the operator begins execution and time it completes execution. [Ref. 16]

Figure 4 shows a segment of the augmented data flow diagram from the ATACMS prototype. Note that the operator "asas_op" has a MET of 200 ms assigned to it. The presence of this MET means that the CAPS scheduler will treated it as a time critical operation. The absence of a MET, such as in operator "lan2_link_op" below, means that the CAPS scheduler will treat the operator as a non-time critical operation. [Ref. 17]

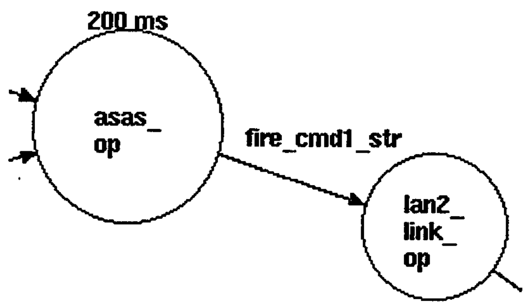
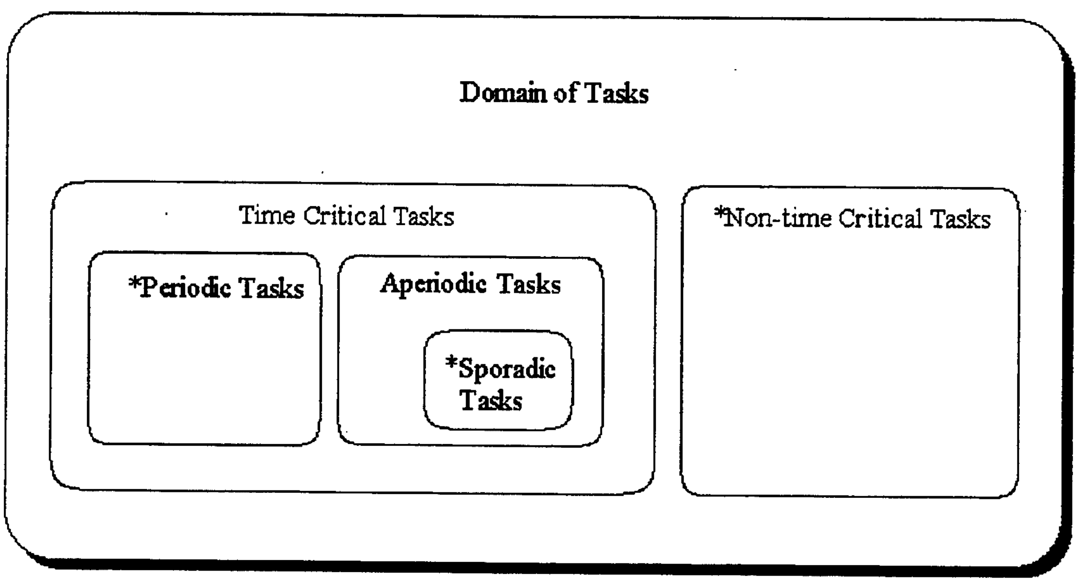


Figure 4. Sample DFD Segment

A further expansion of the domain of tasks begun above yield the Venn diagram in Figure 5. A discussion of time critical tasks follows.

- **Characteristics**

  A time critical task is one that has a timing constraint associated with it. In CAPS, an operator having a MET is considered time critical and will be scheduled in the static schedule loop of the prototype driver task located in the "<prototype_name>.a" file(the prototype driver file). These tasks are considered HIGH priority.



* CAPS available tasks

Figure 5. Domain of Ada Tasks [Ref. 18]

- **Triggering**

There are only two ways to trigger time critical tasks. The first is using a time event and these tasks are called periodic. An example is an airborne phased array radar like on the JSTARS aircraft that scans the battlefield at regular intervals. The second way to trigger a task is with a physical event and these tasks are called aperiodic. An example is using a mouse to press a Quit button on a GUI. A subset of aperiodic tasks are sporadic tasks. The distinction is subtle and will be discussed later. [Ref. 18]

- **Periodic tasks**

Periodic tasks are those which must be accomplished at regularly scheduled intervals (though some variance may be introduced through "jitter"). The requirements analysis will normally drive the size of the period. In a hard real-time system, failure to meet this timing constraint will, by definition, lead to system failure (see Chapter IIF). The CAPS scheduler starts with periodic tasks when it builds the prototype's static schedule.

- **Aperiodic tasks**

These are tasks triggered by some external event such as pressing a mouse button or detecting a hardware interrupt. [Ref. 19] The driver program periodically looks for the triggering event and then executes the task in the required MET. The time between "looks" for the triggering event is the "trigger period" of the aperiodic event. [Ref. 18] It should also be clear that where a normal system may just ignore an event if it happened at a rate faster than the system could execute the triggered event, a real-time system should be built to handle a defined worst case situation. To handle this worst case situation of repetitive inputs without overflow, CAPS incorporates a special sub-type aperiodic task called a sporadic task. Sporadic tasks are aperiodic tasks in which a minimum period between any two aperiodic events is required. [Ref. 19] In CAPS, they are triggered by the arrival of data on data streams.

The following describes the non-time critical tasks:

- **Characteristics**

  A non-time critical task is one that does not have a timing constraint associated with it. In CAPS, an operator without a MET is considered non-time critical and will be scheduled in the dynamic schedule loop of the prototype driver task located in the "<prototype_name>.a" file. These tasks are considered LOW priority and will be executed on a time permitting basis. They execute during periods of availability in the static schedule and are triggered by the arrival of data on a data stream.

Thus, the three operator options available to the CAPS prototype designer are the periodic, sporadic, and non-time critical operators.

H. SPECIFIC METHODOLOGY

This section describes the specific methodology used in analyzing and building the prototype of the ATACMS End-to-End System. This description is provided so the reader may understand the reasoning behind many of the research and design decisions which are described later. Mid-research changes made in the approach will be highlighted as well as the reasons for such changes.

Firstly, we contacted the ODTSE&E to determine potential customers or stakeholders and to determine the technical points of contact who could assist in supplying information or assisting in research. We concluded that the only recognized stakeholder was the tasking office.

The problem of managing a multi-service project such as this one has not been adequately addressed. No existing overall authority was found for developing and disseminating the software requirements and thus the component authorities were, or wished to be, limited in their support. Each element in the system is managed by a separate agent. As best as we could ascertain there is no central database which contains the necessary data to construct a verifiable model. The alternative, detailed interviews with technical personnel familiar with each element, was logistically and programatically unfeasible.

The essence of the problem is the lack of attainable, accurate, and verifiable data of the quantity and quality needed. This is acerbated since, at this stage of the design cycle, the large portion of classified information precludes ascertaining the general requirements needed. The tasking office does not have the resident technical expertise and can only assist in requests for information.

The above impacted the research and model development as follows:

- Required application of resident expertise in defining the requirements.
- Substitution of classified data with replacement data or placeholders clearly identified as such.
- Construction of the model as a tool to study and gain experience in the modeling process vice concentration on useable output.

Using the IBTA, ATACMS performance specifications, and interviews with some of the principals involved, work on a single instance model was begun. The first was a basic model of the system used to verify communications between operators in the system. Refinement I of the model incorporated some of the functions identified during the analysis phase as being essential to modeling the system as well as more appropriate data types. Refinement II involves expanding the model to include other sensors and alternate data paths and is not fully implemented.

The case for a generic model is strong and alluring. However, given the paucity of available data, the decision was made to pursue a single instance of the "golden thread" and then progress to a more general model as time permitted. The selection of the JSTARS to ATACMS thread from the IBTA was due to the relatively good data available for it.

III. REQUIREMENTS MODEL

A. ANALYSIS

1. General Analysis

This chapter describes the systems analysis done on the ATACMS End-to-End System as it pertains to the model within this thesis. A general account of the system requirements is given, followed by a description of the individual iterations of the model. As additional requirements are added, the analysis for those additional requirements are given at that time.

The IBTA vignettes described in Chapter IID seek to identify selected data paths for the various sensor to shooter connections which are most probable and can be used to generally describe the system. For reasons given previously, the basic model represents the JSTARS to ATACMS data path and is outlined in the customer description:

> Intelligence data is collected by the JSTARS, which is linked to a Common Ground Station (CGS). Data from the CGS is provided to the All-Source Analysis System (ASAS) or Advanced Field Artillery Tactical Data System (AFATDS), depending on the mode of operation. In the ASAS, the information is included in a database and made available to authorized users. The DOCC utilizes the AFATDS which uses the targeting related information via a call for fire message. This data is then passed along through the field artillery communications nodes, where it winds up as firing data for an ATACMS equipped unit.

To review, the JSTARS is an airborne radar platform which provides video data to selected ground stations. The Common Ground Station (CGS) is a communications station which is being acquired and which will be able to receive both voice and data from a variety of sources. The ASAS is the location at Corps HQ where the data from the various sources is centrally collected, processed and disseminated. The AFATDS is the fire control system used by the artillery units to communicate and disseminate fire missions.

The Deep Operations Coordination Center (DOCC) is a new element being added to Corps HQ to handle operations beyond the normal corps battlefield. Available information on the DOCC is minimal, so the Corps Tactical Operations Center (CTOC) was substituted since its attributes would likely be very similar.

Using the above vignette and drawing on our experience and knowledge of like systems, the following goals hierarchy was developed. Due to the unavailability or classification of many of the criteria used, we have elected to use a substitute set of numerical requirements. These are represented by the following variable series:

- P_i series denotes required probabilities
- T_i series denotes required time constraints
- E_i series denotes required error limits (percent)

To the IBTA vignette we added a global performance statement:

The system must successfully identify, process, target, and launch within operational time constraints, and success & error rates (T_{total}, P_{total}, E_{total})

From the vignette and global performance statement we developed a goals hierarchy which would guide the design and implementation of the model (Figure 6).

G1 JSTARS will detect enemy targets, and, generate and transmit JSTARS video

G1.1 JSTARS will correctly detect enemy targets with at least P_1 accuracy

G1.2 JSTARS will generate and process video in at least T_1 time.

G1.3 JSTARS will not generate more than E_1 percent errors.

G2 SCDL will transmit the JSTARS video to the GSM

G2.1 SCDL will transmit the JSTARS video with at least P_2 accuracy

G2.2 SCDL will transmit the JSTARS video in at least T_2 time.

G2.3 SCDL will not generate more than E_2 percent errors.

G3 Command Station will receive, process, and act upon the JSTARS video

G3.1 GSM will receive, process and forward the JSTARS video

G3.1.1 GSM will perform with at least P_3 accuracy

G3.1.2 GSM will receive, process, and forward the JSTARS video in at least T_3 time.

G3.1.3 GSM will not generate more than E_3 percent errors.

G3.2 ASAS will process GSM video, identify and prioritize threats, and forward target list

G3.2.1 ASAS will perform with at least P_4 accuracy

G3.2..2 ASAS will receive, process, and forward the JSTARS video in at least T_4 time.

G3.2..3 ASAS will not generate more than E_4 percent errors.

G3.3 CTOC will generate firing command(s) from ASAS target list

G3.3.1 CTOC will perform with at least P_5 accuracy

G3.3.2 CTOC will receive, process, and forward the JSTARS video in at least T_5 time.

G3.3.3 CTOC will not generate more than E_5 percent errors.

G4 CNR will transmit fire missions to the shooter

G4.1 CNR will transmit the fire missions with at least P_6 accuracy

G4.2 CNR will transmit the fire missions in at least T_6 time.

G4.3 CNR will not generate more than E_6 percent errors.

G5 The shooter will fire in response to received fire missions

G5.1 CNR will transmit the fire missions with at least P_6 accuracy

G5.2 CNR will transmit the fire missions in at least T_6 time.

G5.3 CNR will not generate more than E_6 percent errors.

Figure 6. Goals Hierarchy

Normally, when developing the specification for a system, specific consideration is given to the following five constraints on the system:

- Resource - schedule, budget, manpower
- Performance - execution time, accuracy, memory, system down time
- Environment - hardware, operating system, external systems
- Form - Programming language, coding & documentation standards
- Methods - development tools, testing procedures, performance benchmarks [Ref. 5]

However, when using rapid prototyping to quickly identify system requirements and projected performance parameters, many of the above considerations fall out of scope. In general, the focus is on performance and environment, though considerations from the other constraints can be, and often are included. Much of the work is also done at the functional specification level since one of the main purposes of the model is the definition of the external interfaces. The use of representative data types accomplishes much in representing these interfaces, with PSDL attributes used to define the remainder.

2. Components

In this section we will analyze the particular requirements of the individual components. A brief description of the attributes of each of the operators identified in the JSTARS to ATACMS data path are included. These descriptions are a compilation of both known and presumed attributes. Many items listed would be necessary in later refinements to make the system model more robust. Also, for most of these operators it is vital to keep in mind the discussion on quantity of operators given in Chapter IIB. [Ref. 6] [Ref. 7]

- **Target Emitter**
  The model must include an operator which simulates the "real" world in the sense that targets are placed and removed, and they "emit", either

actively or passively, detectable amounts of EM radiation. This is not part of the ATACMS system. Target Emitter would operate at a period sufficient to test the reaction time of the system. It would include in its target generation all the information which could be discerned from the target (i.e., derived attributes) itself by any sensor observing it (currently limited to JSTARS). Since the real world emissions occur instantly this operator would need to function in a short period so as not to disturb the basic behavior of the system

- **JSTARS**
  Since JSTARS generates a continuous series of video frames to be transmitted to receiving ground stations, its operator in the system is periodic. However, given the restrictions of our single processor system, the period needs to be as long as possible (and still meet the system update needs) rather than a continuous cycle. The JSTARS operator has a requirement for functions to generate a video picture, inject simulated errors, and have timing delays to represent operator, processing, and internal transmission time (as opposed to latency).

- **SCDL Link**
  The system contains an automated link between the JSTARS and Command Station modules. Since the link operates upon the receipt of a video frame from JSTARS and must finish within a prescribed time, it is a sporadic operator. The essential attributes of this operator are protocol, baud rate, formats, average message size, and bandwidth (and a derived latency attribute). The link also has a requirement for a function to introduce noise/error into the transmission. Internally, there is no modification of the video picture (exception: noise/error).

- **Ground Station Module**
  The system includes an operator to receive and process the incoming JSTARS video and forward the data to the ASAS on the ASAS LAN. Since the link operates upon the receipt of a video frame from SCDL Link and must finish within a prescribed time, it is a sporadic operator. It is unknown at present what the GSM processing capability is. In our model the GSM will be required to "build" its own video picture for transmission with some defined susceptibility to error. The GSM has requirements for timing delays to represent operator, processing, and internal transmission time (as opposed to latency).

- **ASAS LAN**
  Within the decomposed Command Station operator, all internal communications are accomplished via the ASAS LAN (in our model

the LAN appears between the GSM and the ASAS, and also between the ASAS and the CTOC). Since the link operates upon the transmission of data from a sender, and must finish within a prescribed time, it is a sporadic operator. The essential attributes of this operator are protocol, baud rate, formats, average message size, and bandwidth (and a derived latency attribute). The link also has a requirement for a function to introduce simulated noise/error into the transmission. Internally, there is no modification of the data (EXC.: noise/error). The LAN uses the U.S. Message Text Format (USMTF).

- **ASAS**
  The system includes an operator to represent the ASAS' capability to accept incoming target and intelligence data and perform evaluation and targeting functions. The ASAS determines a priority target list based on a predetermined algorithm and forwards the target list, one target at a time, to the CTOC on the ASAS LAN. Since the ASAS continues to process the target list independent of new data arriving, it is periodic. However, it also contains a "triggered if" condition based on the state stream. The target list generation would have a defined susceptibility to error. The ASAS has requirements for timing delays to represent operator, processing, and internal transmission time (as opposed to latency).

- **CTOC**
  The system includes an operator to receive and process incoming prioritized target lists from the ASAS and generate and forward fire missions to individual batteries on the CNR net. Since the link operates upon the receipt of a new target list from the ASAS and must finish within a prescribed time, it is a sporadic operator. It is unknown at present what the CTOC procedure is. The CTOC has requirements for timing delays to represent operator, processing, and internal transmission time (as opposed to latency).

- **CNR Link**
  The system contains an automated link between the Command Station and Shooter modules. Since the link operates upon the receipt of a firing command from the Command Station, and must finish within a prescribed time, it is a sporadic operator. The essential attributes of this operator are protocol, baud rate, formats, average message size, and bandwidth (and a derived latency attribute). The link also has a requirement for a function to introduce simulated noise/error into the

transmission. Internally, there is no modification of the firing command (exception: noise/error).

- **Shooter**
  The system includes an operator to receive and process the incoming fire missions from the CTOC and generate a "weapons release" against the targeted location. The Shooter needs to reply to the CTOC after a successful launch. Since the shooter operates upon the receipt of a the firing command from the CTOC and must finish within a prescribed time, it is a sporadic operator. As previously mentioned, the behavior of the artillery architecture below Corps is well defined and does not need to be modeled in detail. The Shooter has requirements for timing delays to represent operator, processing, and internal transmission time (as opposed to latency).

The above is a general list of requirements which would need to be verified by the customer during the review process. They are sufficiently general in nature as to cover the top level behavior of the components and to act as flags to the customer as to specific data items which need to be addressed. The operation of the above described system will be in accordance with the goals outlined in the previous section.

B. BASIC MODEL

In this section we discuss the design and construction of the basic model (see graph and PSDL list; Appendix A). As mentioned previously, our purpose was to implement a basic representation of the system and to verify communications between all the components. Several output statements were inserted to allow monitoring of the data propagation. Internal functioning of the system or the suitability of data types was ignored at this stage (all passed data types are the same). For this reason the Ada source code is omitted.

Though the descriptions below are PSDL oriented, the concepts embodied in the individual attributes should be straightforward. If further detail is desired we recommend the CAPS group at NPS be contacted notes be consulted.

- **target_emitter_op**
  This is a periodic operator with a 1 second cycle time, which was selected to contribute to a low system load factor (we wished to keep it below 50%). The target emitter produces an output stream emission_str, which simulates the content of an actual passive or active emission from the target(s). For test purposes, in the basic model the target emitter generates a new "target" stream with each cycle. This new data steram can then be monitored as it propagates downstream. Desiring to limit the impact of this operator on the total system, a 200 ms maximum execution time is given.

- **jstars_op**
  This is a periodic operator with a 1second cycle time, which was selected so as to contribute to a low system load factor (we wished to keep it below 50%). JSTARS accepts the emitter_str and produces an error free output stream, target_array1_str, which simulates the content of the JSTARS video. JSTARS continuously generates its video picture independent of whether the target "picture" has changed. The hard requirement with respect to timing is that JSTARS always operate at least as often as target emitter to preclude the loss of data (in general, JSTARS does not deal with a target set which rapidly changes, see Chapter IIE). Desiring to limit the impact of this operator on the total system, a 200 ms maximum execution time is given.

- **scdl_link_op**
  This is a non-time critical operator which is triggered by the receipt of all target_array1_str from JSTARS. In the basic model the SCDL Link simply passes on the received data and produces an error free output stream, target_array2_str.

- **command_station_op**
  This is a composite operator which represents the system components physically located at the Corps HQ. Its elements are the ground station module (grnd_stat_mod_op), the ASAS LAN connections (lan1_link_op, lan2_link_op), ASAS (asas_op), and the CTOC (ctoc_op).

- **grnd_stat_mod_op**
  This is a non-time critical operator which is triggered by the receipt of all target_array2_str from scdl_link_op. In the basic model the ground station module simply passes on the received data and produces an error free output stream, target_array3_str which is sent to lan1_link_op.

- **lan1_link_op**
  This is a non-time critical operator which is triggered by the receipt of all target_array3_str from grnd_stat_mod_op. In the basic model the ASAS LAN simply passes on the received data and produces an error free output stream, target_array4_str.

- **asas_op**
  This is a periodic operator with a cycle of 4 seconds. In the basic model the ASAS simply passes on the received data and produces an error free output stream, fire_cmd1_str.

- **lan2_link_op**
  This is a non-time critical operator which is triggered by the receipt of all fire_cmd1_str from asas_op. In the basic model the ASAS LAN simply passes on the received data and produces an error free output stream, fire_cmd2_str.

- **ctoc_op**
  This is a non-time critical operator which is triggered by the receipt of all fire_cmd2_str from lan2_link_op. In the basic model the CTOC simply passes on the received data and produces an error free output stream, fire_cmd3_str.

- **cnr_link_op**
  This is a non-time critical operator which is triggered by the receipt of all fire_cmd3_str from command_station_op. In the basic model the CNR Link simply passes on the received data and produces an error free output stream, fire_cmd4_str.

- **shooter_op**
  This is a non-time critical operator which is triggered by the receipt of all fire_cmd4_str from cnr_link_op. In the basic model the shooter simply outputs a "fired" message.

When executed, the basic model runs both the target emitter and the JSTARS based on the static schedule created by CAPS. After running target emitter and JSTARS, the prototype runs each triggered operator in sequence until the data path is complete. As currently timed the basic model completely handles each new "target" prior to generating a new one.

C. REFINEMENT I

In this section we discuss the design and construction of the first refinement of the prototype model (see graph, PSDL list, and Ada source code; Appendix B). The goal of this refined model is to incorporate the attributes and functions which will meet the requirements list delineated earlier. Additionally, a basic GUI has been added that allows the user to control the prototype operation. Since many of the attributes are either unknown or classified, placeholder functions have been used pending identification by the customer and operation of the model in a secure environment.

A bulletized description of the modifications made to the basic model is given for each operator, as well as identification of areas where customer feedback is required. This section should be used as a checklist for user feedback.

- **target_emitter_op**
  - Cycle time increased to 16 seconds.
  - Target generator added (controlled by GUI). User must supply actual target rates and behaviors for testing scenario.
  - Array simulates geographical area under observation by sensor

- **jstars_op**
  - Cycle time increased to 8 seconds.
  - User must supply actual video data type for either use or simulation.
  - Given single processor limitation, user must define a minimum update rate suitable for target volatility
  - Placeholder functions added for simulating operator time, processing time, internal transmission (prep) times, error rate. User must supply actual values.
  - Array simulates video format

- **scdl_link_op**
  - Error injection function added. User must define actual error rates and types.
  - Latency added to data stream (via delay statements). User must supply actual values.

- Placeholder functions added for processing time, internal transmission (prep) times. User must supply actual values. (assumed an automated system, hence no operator time)

- **grnd_stat_mod_op**
  - Grid fixing functionality added. User must supply actual functionality.
  - Target collation function added. User must supply actual functionality.
  - Placeholder functions added for simulating operator time, processing time, internal transmission (prep) times, and error rates and types. User must supply actual values.

- **lan1_link_op**
  - Error injection function added. User must define actual error rates and types.
  - Latency added to data stream (via delay statements). User must supply actual values.
  - Placeholder functions added for processing time, internal transmission (prep) times. User must supply actual values.

- **asas_op**
  - Changed to a periodic operator with a 4 second cycle.
  - Prioritization and targeting functionality added. User must provide actual algorithms and capabilities.
  - Placeholder functions added for simulating operator time, processing time, internal transmission (prep) times, and error rates and types. User must supply actual values.

- **lan2_link_op**
  - Error injection function added. User must define actual error rates and types.
  - Latency added to data stream (via delay statements). User must supply actual values.
  - Placeholder functions added for processing time, internal transmission (prep) times. User must supply actual values.

- **ctoc_op**
  - Currently passes through fire mission. User must provide actual algorithms and capabilities.
  - Placeholder functions added for simulating operator time, processing time, internal transmission (prep) times, and error rates and types. User must supply actual values.

- **cnr_link_op**
  - Error injection function added. User must define actual error rates and types.
  - Latency added to data stream (via delay statements). User must supply actual values.
  - Placeholder functions added for processing time, internal transmission (prep) times. User must supply actual values. (assumed an automated system, hence no operator time)

- **shooter_op**
  - Fire upon valid command functionality added. User must provide additional algorithms and capabilities.
  - Placeholder functions added for simulating operator time, processing time, internal transmission (prep) times, error rate. User must supply actual values.

Additional requirements identified in this refinement of the model are the overall system timing and accuracy constraints as well as the defined failure modes. Specifics are not currently incorporated since many of these items are either unknown or classified. Specific data must be supplied by the user.

One area which presented particular difficulty was the insertion of data stream latency. Under particular circumstances of time critical operators, long latencies, and short execution times/periods, it becomes impossible to generate a valid schedule for the prototype driver program. In lieu of using the latency feature in CAPS, we chose to insert delay statements into the appropriate operators. Currently this generates, as expected, a series of timing errors. These errors can be isolated in code from errors generated by other causes.

The basic GUI consists of a pair of TAE generated control panels. The first, Target Panel controls target_emitter_op and enables the human operator to run, pause and quit the prototype as well as initiate target generation. The Shooter Fire Mission Panel displays vital target data as it exists after being processed and transmitted through the system. It also tests and displays the communications status as determined by the receipt of a flag signifying a failure of the system communications path.

Since the purpose of the prototype is to ascertain and verify requirements and less so as a simulation, the use of script files and text output windows were heavily used to observe and document the behavior of the model.

D. REFINEMENT II

The model presented in this section represents our vision for a more robust whole system model which incorporates several elements and data paths not included in Refinement I. This model is not an executable prototype and is intended only to show, in coarse terms, how a model incorporating multiple sensors, varied links, and a more detailed command center might appear.

Several of the operators used in Refinement II are not detailed in either the system description or system analysis. Lack of data prevented us from using some of these operators earlier. For instance, the Tactical Information Broadcast System (TIBS) description seems to indicate some interesting behaviors and attributes which would not be modeled in any of the other links. TIBS requires a sensor input, hence the appearance of Rivet Joint, though no details on it are available at this time. Also, we included the Area Common User System (ACUS) which was a necessary adjunct to Trackwolf, both of which we wished included.

A description of the modifications and additions to Refinement I resulting in Refinement II are detailed below, including reasons for selection:

- **grcs_op**

  - This operator represents the Guardrail Common Sensor (GRCS). Since it is a SIGINT sensor its operations, data type and content, and thus behavior, would be different from JSTARS.
  - While there is no dedicated link for GRCS, some of the link attributes would pertain to the GRCS transmission and thus would need to be represented in either grcs_op or cgs_op.

- **rivet_joint_op**

  - This operator represents the Rivet Joint airborne sensor. No information concerning Rivet Joint was available to us.
  - Rivet Joint utilizes TIBS as a link to the Commander's Tactical Terminal (CTT).

- **Trackwolf**

  - This operator represents the Trackwolf sensor. This sensor's utility in targeting deep targets makes its inclusion in the model desirable
  - Trackwolf utilizes the ACUS.

- **uav_op**

  - This operator represents ground launched unmanned aerial reconnaissance vehicles. Since it provides raw video, its data type and content, and thus behavior, would be different from JSTARS.
  - While there is no dedicated link for the UAV, some of the link attributes would pertain to the UAV video and thus would need to be represented in either uav_op or cgs_op.

- **tibs_op**

  - This operator represents the TIBS. The system appears to have some data processing and fusion capabilities, thus presenting unique modeling issues.

- **acus_op**

  - This operator represents the ACUS communication net. It appears to be a relatively open WAN and thus also presents some unique modeling issues.

Once a thorough customer review of Refinement I occurs and the multitude of open issues in that model are addressed, Refinement II can be used as a launching point for expansion of the ATACMS model.

IV. CONCLUSIONS AND RECOMMENDATIONS

A. EVALUATION OF MODEL

Evaluation of the prototype model constructed is limited to Refinement I. The Basic Model was used only as a starting point to verify some initial relationships and data flows. Refinement II is simply an expansion of its predecessor and represents preparatory work in advance of a review with the customer. Refinement II does not address any issues concerning accuracy and suitability of either the analysis or the prototype not raised by Refinement I. Refinement II incorporates a single additional operating data flow which was used to attempt to quantify CAPS productivity (see next section).

We believe that Refinement I accurately represents the primary relationships between the operators identified by the analysis. Secondary and tertiary data flow paths, such as those used for back-up or streamlined operations have neither been identified nor incorporated. The prototype is ready for its first evaluation by the tasking office or their representatives. The issues which would enable progression to the next level of accuracy and suitability have been identified. The detailed discussions of the requirements models in the preceding chapter may be used as a preliminary checklist to begin the evaluation process.

The prototype as currently configured will not provide an accurate appraisal of the operational behavior of the system. Placement of the actual data values into the both the placeholder function and the substitution data we provided is necessary prior to evaluation of the prototype operationally.

Visibility into the model should be sufficient at this stage to do a preliminary evaluation once the actual data is substituted. Further use of script files and screen outputs may be useful in the next phase to be able to rigorously verify the prototype. We found that limiting the GUI to only essential items and avoiding the more robust graphics significantly reduced the impact on the model. Ultimately this approach ensures a more accurate evaluation as the prototype grows. While this may not be a critical issue for "soft" real-time systems, "hard" systems demand that non-system operators (e.g. GUI, model control, etc.) minimize their influence on the actual system operators.

Once the model is updated and verified for the JSTARS to ATACMS thread, it will be a relatively simple matter to expand the model to encompass other sensor and weapons, as well as alternate data flows by employing widespread reuse of the operators and their common contents.

B. EVALUATION OF CAPS

An evaluation of CAPS as a requirements specification tool must necessarily be a subjective process for the authors. It is difficult to differentiate limitations and shortcomings imposed by the CAPS environment from those attributable to either the paucity of data, lack of actual systems analysis experience, or the learning curve necessary to become proficient in the use of the system.

From the viewpoint of rigor, CAPS is able, either directly or through derived attributes, to represent every requirement drawn from the resource materials with the notable exception of the data stream latency noted in Refinement I of the model. The automatic scheduling, graphical translation, and code generation were essentially error free (a pair of one-line changes required for each prototype driver generated currently require manual correction). Management tools allow the smooth transition between versions and provide tight control of the development process. In fact, we found that parallel efforts on small portions of the model were possible. We recommend the use and evaluation of the database management system on a larger, more detailed effort. Of a special interest is the ability to exclude portions of the code from various members of a development team, thus supporting tight configuration management.

While the CAPS system appears generally complete in its functional capabilities, it does present some areas for improvement. We feel the largest of these is the lack of documentation of system features and procedures. While fully aware that CAPS has heretofore been largely in the development stage, breakout into widespread use is going to be very dependent on the availability of useful and complete documentation of the system. Without the availability of publications or on-line *HELP*, it would not take a very large installed base to quickly swamp the CAPS group's ability to respond to user inquiries. Also, managers will demand a more rapid ramp up than is currently possible without such information.

We have attempted to begin the process of collecting and developing a thorough database for developing the documentation required. Detailed notes were kept throughout the process, documenting esoteric, erroneous, or difficult to use procedures, commands and processes. A collection of mini tutorials have been include in Appendix D.

Though we were not limited in our effort, due to the nature and size of our problem domain, operating CAPS in a single processor environment poses challenges which would be alleviated by completing work on a multi-platform version.

Platform portability, while not a problem for us, is necessary for the widespread use of the system.

For the immediate future the esoteric nature of some the CAPS procedures, commands, and processes does not disqualify it from use on an actual project. This is so for two major reasons. Firstly, the distribution and use of CAPS on a wider basis would present us with the ability to identify and quantify these issues, as well as user preferences, in much the way a beta release of commercial software does. By this process of user feedback, we naturally migrate towards a common interface model. Secondly, we foresee the benefits resulting from the continuing use of CAPS by a modeling concern as mitigating and recouping the start-up effort the system requires.

We also recommend the complete integration of the TAE multi-file approach. Until recently, prototypes that used a TAE generated interface utilized what is referred to as the "single file" approach. In the single file approach, TAE generates one central file that contains the event loop and procedures that link to all TAE windows. Since CAPS generates its own main program event loop, the TAE event loop must be subjugated to it. The conversion of the TAE driver is a manual 31 step process which needs to be redone each time <u>any</u> item on the interface is changed. Minor changes require large recompilations. The single file paradigm is very monolithic and hard to manage., and it does not lend itself to decomposition. In the TAE multi-file approach, TAE creates a panel package for each TAE panel created. Several automated aids have been developed so that with the incorporation of the multi-file approach, the graphical interface is less of an impedance to the rapid development of a (especially a large) prototype. In our estimation, this has been the single most important productivity enhancement to CAPS to have been incorporated during the time of our research.

With respect to productivity, two significant features demand mention. Firstly, is the automatic driver code generation. The productive impact of the ability to quickly generate error free drivers would require a separate analysis beyond our scope. However, we estimate that the driver code for even a small project would require a programmer with superior knowledge of Ada tasks several weeks to schedule, write, compile and debug. Large system prototyping without automatic code generation simply becomes economically unfeasible. Secondly, is the ability to incorporate rapidly changing design demands (the essence of prototyping). We performed a small exercise where the CAPS user was given a requirements change without notice and the result was timed. We incorporated changes in five of the operators, converting them to time-critical types. This required the regeneration of the operator schedule and prototype driver program. Total effort was 2.5 hours. We estimate that an identical effort done manually would take at least 12 hours (2 hours to develop the schedule, 8 hours to write the changes to the driver program, and 2 hours to debug) . Had additional operators been required, an even more pronounced difference would be observed.

To reiterate, we find no "show-stoppers" in the use of CAPS as a requirements generation tool. We highly recommend that the next logical phase be undertaken: to insert CAPS into a real world requirements development effort with a team of experience systems analysts to evaluate it.

C. UNRESOLVED ISSUES

The major unresolved issue with respect to the requirements specification and the model is the first of what would have to be several customer evaluations and verifications. This is made problematic by a second issue, that there is no cognizant technical authority (or convened body) to evaluate the model or guide us in these further iterations. The lack of a program "sponsor" who can direct or authorize the use of CAPS as an integral part of the development team constrains both our work and any possible follow on effort.
We would also like to see the requirements utilize more generic definitions, increasing its utility and range of application

With respect to the CAPS tool itself, we are still uncertain whether CAPS is ready for widespread distribution as an alternative to current products and processes in place which have clear shortcomings. The application of CAPS in a beta environment as a (the) primary tool for requirements generation by a team of experienced system analysts is a must. Secondly, continued modification of the CAPS' environment to make it easier to learn and use, as well as increasing productivity is necessary, even if parallel with the beta effort. Those must include a major effort at development of a set of references (publication or on-line). Platform portability will be crucial, and a study of current platforms in the installed base for other tools should be conducted. We must make CAPS more user friendly.

Modification of the schedule engine to address the shortcomings associated with the data stream latencies should be a high technical priority. Without incorporation of these changes, the prototype developer must resort to a series of less than optimum work-around.

The following is a summary of suggested projects or areas of research with a first cut estimate of labor requirements based on our experience with CAPS and like efforts within the CAPS' group. The use of NPS graduate students for these efforts, while cutting costs, would entail a 1.5-2.0 factor for calendar time, since a significant portion of their time is assigned to other tasks:

Development of CAPS System Documentation	.5 years
Upgraded Integrated Development Environment	1.0 years
Setup, Training, and Support of a One Year Requirements Project (including post-project economic evaluation)	.5 years
Platform Portability Study	.25 years
Platform Porting (each; some may not be practical)	.2 - 1.0 years

D. SUMMARY

In summary, a refined ATACMS model is complete and available for review by Office of the Director, Test, Systems Engineering & Evaluation (ODTSE&E) or any designated representative. We believe that with modification of the model to incorporate actual/classified data that the model is sufficiently robust to assist in identifying those Critical Operational Issues outlined in the Memorandum of Agreement covering the ATACMS End-to-End System.

The comprehensive use of CAPS in modeling a real world system has confirmed the essential qualities of the system while identifying several issues for future enhancements. CAPS represents a significant opportunity for DOD to address those software development issues resulting from shortcomings in the requirements process.

LIST OF REFERENCES

1. Office Of the Director, Test, Systems Engineering & Evaluation, *Army Tactical Missile System (ATACMS) memorandum of Agreement (MOA) Status Report—Information Memorandum,* Washington, DC, 26 April 1995.

2. Office Of the Director, Test, Systems Engineering & Evaluation, *ATACMS Draft MOA; Comments on,* Washington, DC, 9 Jun 1995.

3. Deputy Under Secretary of the Army (OR), Assistant Secretary of the Army , Director, Operational Test and Evaluation, and Director, Test, Systems Engineering & Evaluation (USD(A&T)), *Memorandum Of Agreement,* Washington, DC, 9 Jun 1995.

4. Holly, COL, US Army, *Electronic Mail To LCDR Angrisani USN,* Huntsville, AL, Feb 1996.

5. Berzins & Luqi, *Software Engineering with Abstractions: An Integrated Approach to Software Development using Ada,* Addison-Wesley Publishing Company, Reading, MA, 1991.

6. US Army, Combined Arms Command, *Integrated Battlefield Targeting Architecture Handbook, Coordinating Draft,* Fort Leavenworth, KS, 1 May 1994.

7. US Army, Combined Arms Command, *Integrated Battlefield Targeting Architecture Handbook, Architecture Annex, Final Coordinating Draft,* Fort Leavenworth, KS, 1 May 1994.

8. Stankovic, J. A., "Misconceptions about Real-Time Computing. A Serious Problem for Next-Generation Systems", *Computer*, Oct. 1988.

9. Laplante, P. A., "Real-Time Systems Design and Analysis", Piscataway, New Jersey: *IEEE Press*, 1993.

10. Ripps, D. L., *An Implementation Guide to Real-Time Programming.* Englewood Cliffs, New Jersey: Yourdon Press Computing Series, 1989.

11. Gillies, D., *FAQ page for the Real-Time Usenet Group,* http://www.cis.ohio-state.edu/hypertext/faq/usenet/realtime-computing/faq/faq-doc-4.html.

12. Eagle, C.S., Tools For Storage And Retrieval of ADA Software Components In A Software Base, Monterey, CA, *Naval Postgraduate School*, 1995.

13. Luqi, "Software Evolution Through Rapid Prototyping", Piscataway, NJ, *IEEE Press*, 1989.

14. CAPS Research Group, *CAPS Home Page*, http://wwwcaps.cs.nps.navy.mil.

15. Berzins, Luqi, Shing, Computer Aided Prototyping System (CAPS), *Proceedings of the 7th International Conference on Software Engineering and Knowledge Engineering*, Rockville, MD 22-24 June 1995.

16. Luqi, "Computer-Aided Prototyping for a Command-and-Control System Using CAPS", *IEEE Software*, Jan 1992.

17. Luqi, *Class Notes for CS4920*, Naval Postgraduate School, April 1996.

18. M. Shing, *Interview with Professor ManTak Shing by Major George Whitbeck, USMC*, Naval Postgraduate School, August 1996.

19. Cordeiro, *Distributed Hard Real-Time Scheduling for a Software Prototyping Environment*, Ph.D. Dissertation, Naval Postgraduate School, March 1995.

APPENDIX A. BASIC MODEL GRAPH AND PSDL

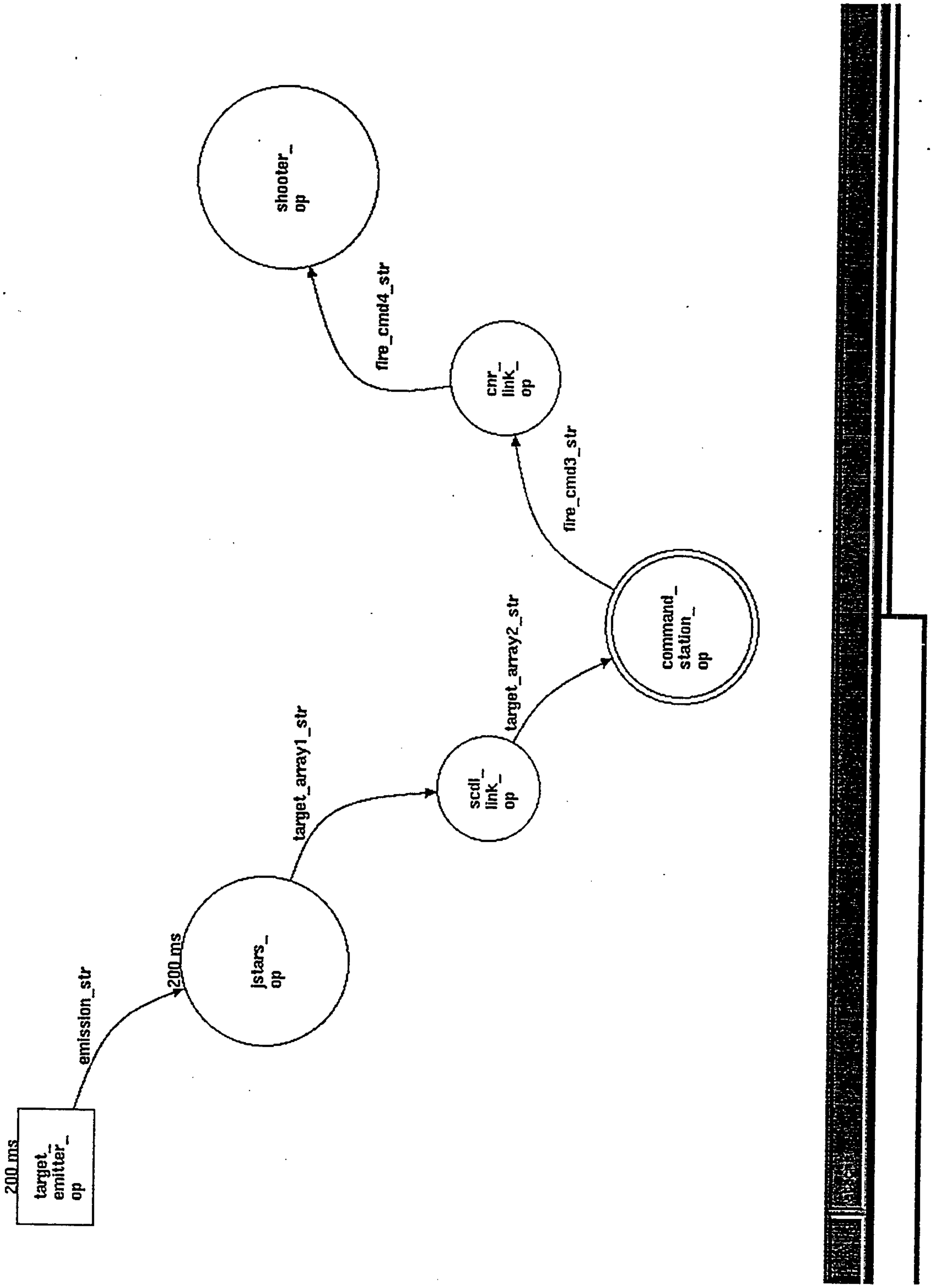


Figure 1. Basic Model - Top Level.

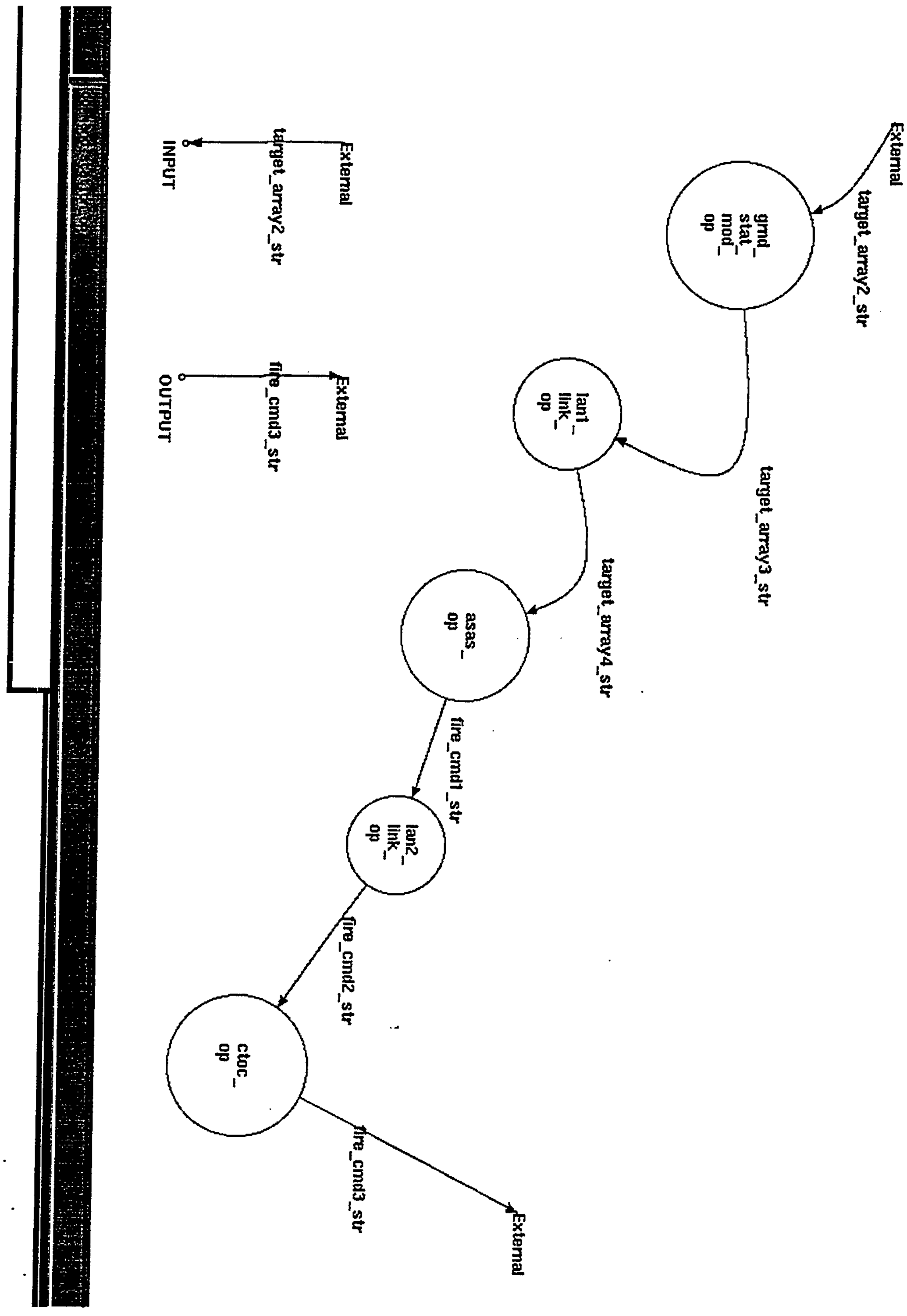


Figure 2. Basic Model - Decomposed Command Station.

BASIC MODEL PSDL

```
TYPE target_data
  SPECIFICATION
END
IMPLEMENTATION ADA target_data

END

OPERATOR asas_op
  SPECIFICATION
    INPUT
      target_array4_str : target_data
    OUTPUT
      fire_cmd1_str : target_data
END
IMPLEMENTATION ADA asas_op

END

OPERATOR atacms
  SPECIFICATION
END
IMPLEMENTATION
  GRAPH
    VERTEX cnr_link_op

    VERTEX command_station_op

    VERTEX jstars_op : 200 MS

    VERTEX scdl_link_op

    VERTEX shooter_op

    VERTEX target_emitter_op : 200 MS

    EDGE emission_str
      target_emitter_op ->
      jstars_op

    EDGE fire_cmd3_str
      command_station_op ->
      cnr_link_op

    EDGE fire_cmd4_str
      cnr_link_op ->
      shooter_op

    EDGE target_array1_str
      jstars_op ->
      scdl_link_op

    EDGE target_array2_str
      scdl_link_op ->
      command_station_op
  DATA STREAM
    emission_str : target_data,
    fire_cmd3_str : target_data,
    fire_cmd4_str : target_data,
    target_array1_str : target_data,
    target_array2_str : target_data
  CONTROL CONSTRAINTS
    OPERATOR cnr_link_op
      TRIGGERED BY SOME
        fire_cmd3_str
```

```
OPERATOR command_station_op

  OPERATOR jstars_op
    PERIOD 1000 MS

  OPERATOR scdl_link_op
    TRIGGERED BY SOME
      target_array1_str

  OPERATOR shooter_op
    TRIGGERED BY SOME
      fire_cmd4_str

  OPERATOR target_emitter_op
    PERIOD 1000 MS
END

OPERATOR cnr_link_op
  SPECIFICATION
    INPUT
      fire_cmd3_str : target_data
    OUTPUT
      fire_cmd4_str : target_data
END
IMPLEMENTATION ADA cnr_link_op

END

OPERATOR command_station_op
  SPECIFICATION
    INPUT
      target_array2_str : target_data
    OUTPUT
      fire_cmd3_str : target_data
END
IMPLEMENTATION
  GRAPH
    VERTEX asas_op

    VERTEX ctoc_op

    VERTEX grnd_stat_mod_op

    VERTEX lan1_link_op

    VERTEX lan2_link_op

    EDGE fire_cmd1_str
      asas_op ->
      lan2_link_op

    EDGE fire_cmd2_str
      lan2_link_op ->
      ctoc_op

    EDGE fire_cmd3_str
      ctoc_op ->
      EXTERNAL

    EDGE target_array2_str
      EXTERNAL ->
      grnd_stat_mod_op

    EDGE target_array3_str
      grnd_stat_mod_op ->
      lan1_link_op
```

```
    EDGE target_array4_str
      lan1_link_op ->
      asas_op
  DATA STREAM
    fire_cmd1_str : target_data,
    fire_cmd2_str : target_data,
    target_array3_str : target_data,
    target_array4_str : target_data
  CONTROL CONSTRAINTS
    OPERATOR asas_op
      TRIGGERED BY SOME
        target_array4_str

    OPERATOR ctoc_op
      TRIGGERED BY SOME
        fire_cmd2_str

    OPERATOR grnd_stat_mod_op
      TRIGGERED BY SOME
        target_array2_str

    OPERATOR lan1_link_op
      TRIGGERED BY SOME
        target_array3_str

    OPERATOR lan2_link_op
      TRIGGERED BY SOME
        fire_cmd1_str
END

OPERATOR ctoc_op
  SPECIFICATION
    INPUT
      fire_cmd2_str : target_data
    OUTPUT
      fire_cmd3_str : target_data
END
IMPLEMENTATION ADA ctoc_op

END

OPERATOR grnd_stat_mod_op
  SPECIFICATION
    INPUT
      target_array2_str : target_data
    OUTPUT
      target_array3_str : target_data
END
IMPLEMENTATION ADA grnd_stat_mod_op

END

OPERATOR jstars_op
  SPECIFICATION
    INPUT
      emission_str : target_data
    OUTPUT
      target_array1_str : target_data
    MAXIMUM EXECUTION TIME 200 MS
END
IMPLEMENTATION ADA jstars_op

END

OPERATOR lan1_link_op
```

```
  SPECIFICATION
    INPUT
      target_array3_str : target_data
    OUTPUT
      target_array4_str : target_data
END
IMPLEMENTATION ADA lan1_link_op

END

OPERATOR lan2_link_op
  SPECIFICATION
    INPUT
      fire_cmd1_str : target_data
    OUTPUT
      fire_cmd2_str : target_data
END
IMPLEMENTATION ADA lan2_link_op

END

OPERATOR scdl_link_op
  SPECIFICATION
    INPUT
      target_array1_str : target_data
    OUTPUT
      target_array2_str : target_data
END
IMPLEMENTATION ADA scdl_link_op

END

OPERATOR shooter_op
  SPECIFICATION
    INPUT
      fire_cmd4_str : target_data
END
IMPLEMENTATION ADA shooter_op

END

OPERATOR target_emitter_op
  SPECIFICATION
    OUTPUT
      emission_str : target_data
    MAXIMUM EXECUTION TIME 200 MS
END
IMPLEMENTATION ADA target_emitter_op

END
```

APPENDIX B. REFINEMENT I GRAPH, PSDL, ADA SOURCE CODE

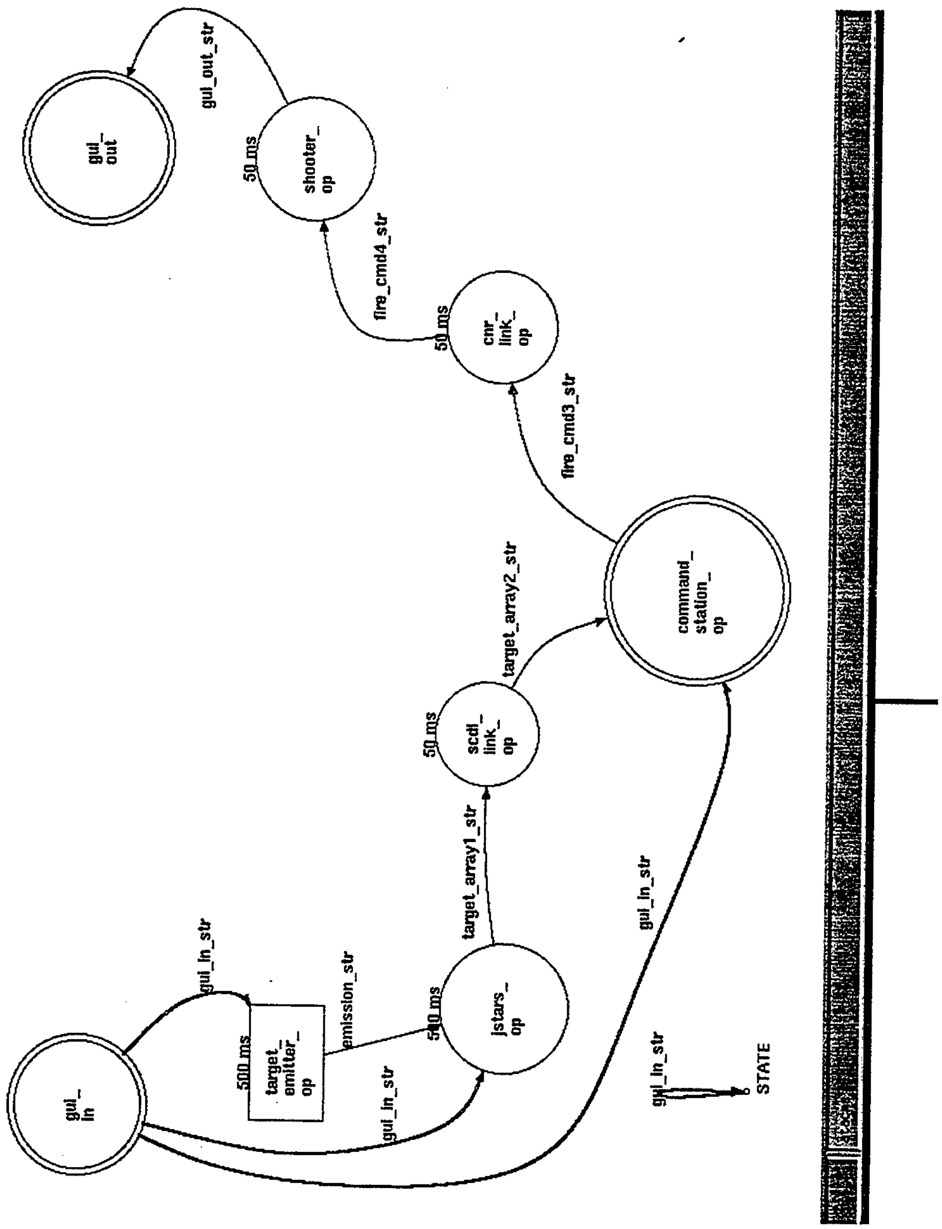


Figure 1. Refinement I Model - Top Level.

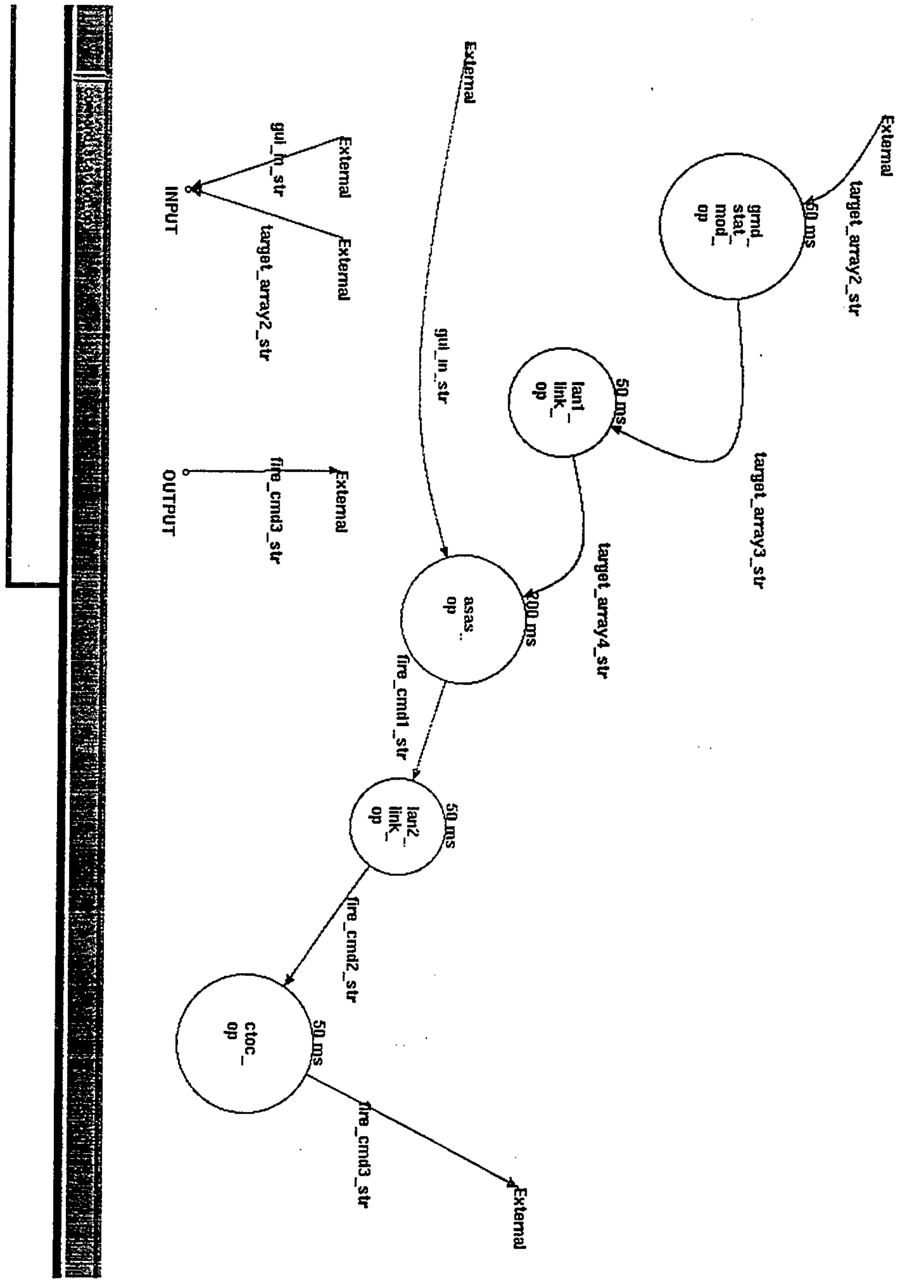


Figure 2. Refinement I Model - Decomposed Command Station.

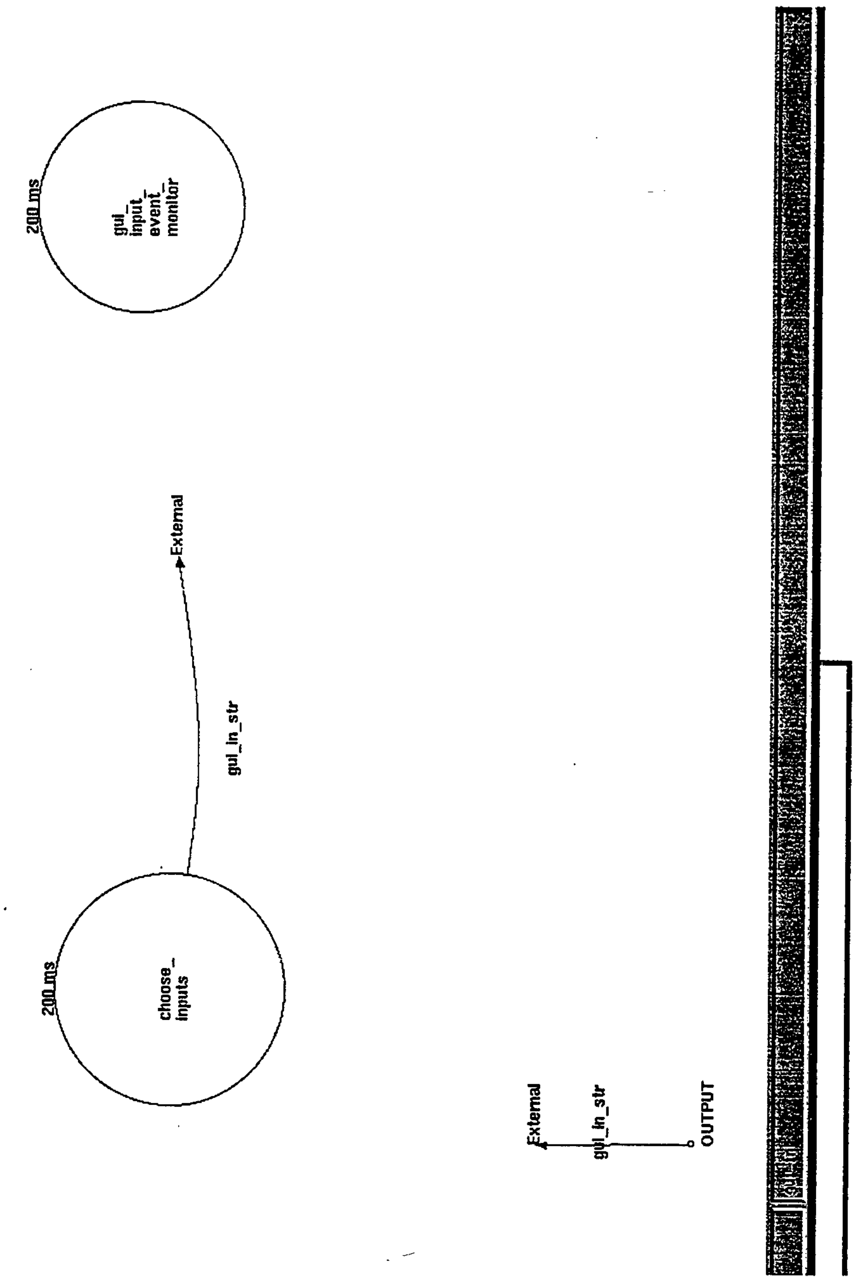
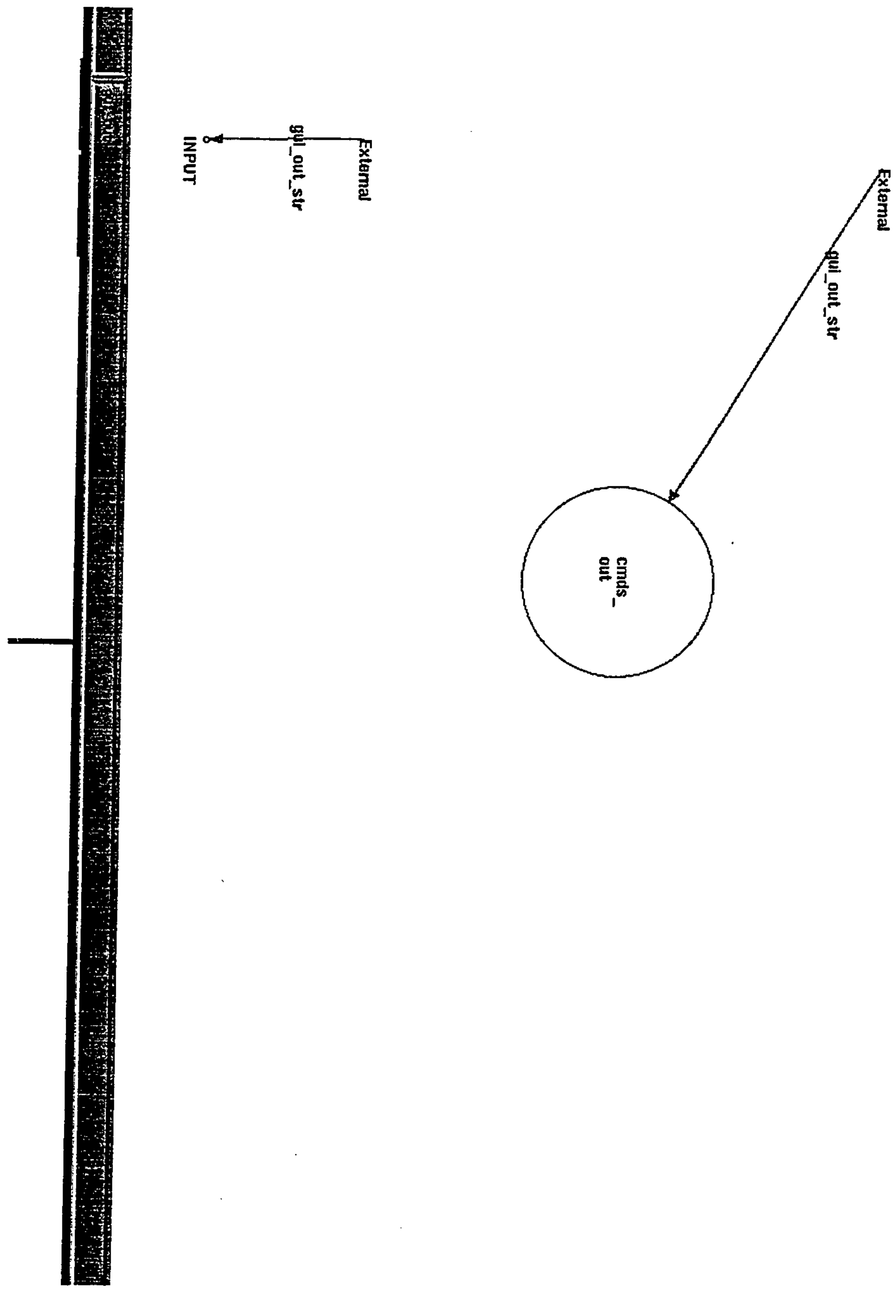


Figure 3. Refinement I Model - Decomposed Gui_in.



Refinement I Model - Decomposed Gui_out.

Refinement I Source Code

atacms.psdl

```
TYPE grnd_stat_mod_array
 SPECIFICATION
END
IMPLEMENTATION ADA
grnd_stat_mod_array

END

TYPE jstars_array
 SPECIFICATION
END
IMPLEMENTATION ADA jstars_array

END

TYPE my_unit
 SPECIFICATION
  OPERATOR pause
   SPECIFICATION
    OUTPUT
     x : my_unit
  END
END
IMPLEMENTATION ADA my_unit

END

TYPE target_data
 SPECIFICATION
END
IMPLEMENTATION ADA target_data

END

TYPE target_emitter_array
 SPECIFICATION
END
IMPLEMENTATION ADA
target_emitter_array

END

OPERATOR asas_op
 SPECIFICATION
  INPUT
   gui_in_str : my_unit,
   target_array4_str :
grnd_stat_mod_array
  OUTPUT
   fire_cmd1_str : target_data
  MAXIMUM EXECUTION TIME
200 MS
END
IMPLEMENTATION ADA asas_op

END

OPERATOR atacms
 SPECIFICATION
  STATES
   gui_in_str : my_unit
   INITIALLY
    pause
END
IMPLEMENTATION
 GRAPH
  VERTEX cnr_link_op : 50 MS

  VERTEX command_station_op

  VERTEX gui_in

  VERTEX gui_out

  VERTEX jstars_op : 500 MS

  VERTEX scdl_link_op : 50 MS

  VERTEX shooter_op : 50 MS

  VERTEX target_emitter_op : 500
MS

  EDGE emission_str
   target_emitter_op ->
   jstars_op

  EDGE fire_cmd3_str
   command_station_op ->
```

```
  cnr_link_op

 EDGE fire_cmd4_str
  cnr_link_op ->
  shooter_op

 EDGE gui_in_str
  gui_in ->
  command_station_op

 EDGE gui_in_str
  gui_in ->
  target_emitter_op

 EDGE gui_in_str
  gui_in ->
  jstars_op

 EDGE gui_out_str
  shooter_op ->
  gui_out

 EDGE target_array1_str
  jstars_op ->
  scdl_link_op

 EDGE target_array2_str
  scdl_link_op ->
  command_station_op
DATA STREAM
 emission_str : target_emitter_array,
 fire_cmd3_str : target_data,
 fire_cmd4_str : target_data,
 gui_out_str : target_data,
 target_array1_str : jstars_array,
 target_array2_str : jstars_array
CONTROL CONSTRAINTS
 OPERATOR cnr_link_op
  TRIGGERED BY SOME
   fire_cmd3_str

 OPERATOR command_station_op

 OPERATOR gui_in

 OPERATOR gui_out

 OPERATOR jstars_op
  TRIGGERED IF
   gui_in_str /= my_unit.pause
  PERIOD 8000 MS

 OPERATOR scdl_link_op
  TRIGGERED BY SOME
   target_array1_str

 OPERATOR shooter_op
  TRIGGERED BY SOME
   fire_cmd4_str

 OPERATOR target_emitter_op
  TRIGGERED IF
   gui_in_str /= my_unit.pause
  PERIOD 16000 MS
END

OPERATOR choose_inputs
 SPECIFICATION
  OUTPUT
   gui_in_str : my_unit
  MAXIMUM EXECUTION TIME
200 MS
END
IMPLEMENTATION ADA
choose_inputs

END

OPERATOR cmds_out
 SPECIFICATION
  INPUT
   gui_out_str : target_data
END
IMPLEMENTATION ADA cmds_out

END

OPERATOR cnr_link_op
 SPECIFICATION
  INPUT
   fire_cmd3_str : target_data
  OUTPUT
   fire_cmd4_str : target_data
  MAXIMUM EXECUTION TIME 50
MS
END
IMPLEMENTATION ADA
cnr_link_op

END

OPERATOR command_station_op
```

```
  SPECIFICATION
   INPUT
    gui_in_str : my_unit,
    target_array2_str : jstars_array
   OUTPUT
    fire_cmd3_str : target_data
END
IMPLEMENTATION
 GRAPH
  VERTEX asas_op : 200 MS

  VERTEX ctoc_op : 50 MS

  VERTEX grnd_stat_mod_op : 50 MS

  VERTEX lan1_link_op : 50 MS

  VERTEX lan2_link_op : 50 MS

  EDGE fire_cmd1_str
   asas_op ->
   lan2_link_op

  EDGE fire_cmd2_str
   lan2_link_op ->
   ctoc_op

  EDGE fire_cmd3_str
   ctoc_op ->
   EXTERNAL

  EDGE gui_in_str
   EXTERNAL ->
   asas_op

  EDGE target_array2_str : 5000 MS
   EXTERNAL ->
   grnd_stat_mod_op

  EDGE target_array3_str
   grnd_stat_mod_op ->
   lan1_link_op

  EDGE target_array4_str
   lan1_link_op ->
   asas_op
 DATA STREAM
  fire_cmd1_str : target_data,
  fire_cmd2_str : target_data,
  target_array3_str : grnd_stat_mod_array,
  target_array4_str : grnd_stat_mod_array

 CONTROL CONSTRAINTS
  OPERATOR asas_op
   TRIGGERED IF
    gui_in_str /= my_unit.pause
   PERIOD 4000 MS

  OPERATOR ctoc_op
   TRIGGERED BY SOME
    fire_cmd2_str

  OPERATOR grnd_stat_mod_op
   TRIGGERED BY SOME
    target_array2_str
   MAXIMUM RESPONSE TIME
5050 MS

  OPERATOR lan1_link_op
   TRIGGERED BY SOME
    target_array3_str

  OPERATOR lan2_link_op
   TRIGGERED BY SOME
    fire_cmd1_str
END

OPERATOR ctoc_op
 SPECIFICATION
  INPUT
   fire_cmd2_str : target_data
  OUTPUT
   fire_cmd3_str : target_data
  MAXIMUM EXECUTION TIME 50
MS
END
IMPLEMENTATION ADA ctoc_op

END

OPERATOR grnd_stat_mod_op
 SPECIFICATION
  INPUT
   target_array2_str : jstars_array
  OUTPUT
   target_array3_str :
grnd_stat_mod_array
  MAXIMUM EXECUTION TIME 50
MS
END
IMPLEMENTATION ADA
grnd_stat_mod_op
```

```
END

OPERATOR gui_in
 SPECIFICATION
  OUTPUT
   gui_in_str : my_unit
END
IMPLEMENTATION
 GRAPH
  VERTEX choose_inputs : 200 MS

  VERTEX gui_input_event_monitor : 200 MS

  EDGE gui_in_str
   choose_inputs ->
   EXTERNAL
 CONTROL CONSTRAINTS
  OPERATOR choose_inputs
   PERIOD 2000 MS

  OPERATOR gui_input_event_monitor
END

OPERATOR gui_input_event_monitor
 SPECIFICATION
  MAXIMUM EXECUTION TIME 200 MS
END
IMPLEMENTATION ADA
gui_input_event_monitor

END

OPERATOR gui_out
 SPECIFICATION
  INPUT
   gui_out_str : target_data
END
IMPLEMENTATION
 GRAPH
  VERTEX cmds_out

  EDGE gui_out_str
   EXTERNAL ->
   cmds_out
 CONTROL CONSTRAINTS
  OPERATOR cmds_out
   TRIGGERED BY SOME
    gui_out_str
END

OPERATOR jstars_op
 SPECIFICATION
  INPUT
   emission_str : target_emitter_array,
   gui_in_str : my_unit
  OUTPUT
   target_array1_str : jstars_array
  MAXIMUM EXECUTION TIME
500 MS
END
IMPLEMENTATION ADA jstars_op

END

OPERATOR lan1_link_op
 SPECIFICATION
  INPUT
   target_array3_str :
grnd_stat_mod_array
  OUTPUT
   target_array4_str :
grnd_stat_mod_array
  MAXIMUM EXECUTION TIME 50
MS
END
IMPLEMENTATION ADA
lan1_link_op

END

OPERATOR lan2_link_op
 SPECIFICATION
  INPUT
   fire_cmd1_str : target_data
  OUTPUT
   fire_cmd2_str : target_data
  MAXIMUM EXECUTION TIME 50
MS
END
IMPLEMENTATION ADA
lan2_link_op

END

OPERATOR scdl_link_op
 SPECIFICATION
  INPUT
   target_array1_str : jstars_array
  OUTPUT
   target_array2_str : jstars_array
  MAXIMUM EXECUTION TIME 50
MS
```

```
END
IMPLEMENTATION ADA scdl_link_op

END

OPERATOR shooter_op
 SPECIFICATION
  INPUT
   fire_cmd4_str : target_data
  OUTPUT
   gui_out_str : target_data
  MAXIMUM EXECUTION TIME 50 MS
END
IMPLEMENTATION ADA shooter_op

END

OPERATOR target_emitter_op
 SPECIFICATION
  INPUT
   gui_in_str : my_unit
  OUTPUT
   emission_str : target_emitter_array
  MAXIMUM EXECUTION TIME 500 MS
END
IMPLEMENTATION ADA target_emitter_op

END
```

atacms.a

```
package ATACMS_EXCEPTIONS is
  -- PSDL exception type declaration
  type PSDL_EXCEPTION is (UNDECLARED_ADA_EXCEPTION);
end ATACMS_EXCEPTIONS;

package ATACMS_INSTANTIATIONS is
  -- Ada Generic package instantiations

end ATACMS_INSTANTIATIONS;

  with PSDL_TIMERS;
package ATACMS_TIMERS is
  -- Timer instantiations
end ATACMS_TIMERS;

-- with/use clauses for atomic type packages
  with GRND_STAT_MOD_ARRAY_PKG; use GRND_STAT_MOD_ARRAY_PKG;
  with JSTARS_ARRAY_PKG; use JSTARS_ARRAY_PKG;
  with MY_UNIT_PKG; use MY_UNIT_PKG;
  with TARGET_DATA_PKG; use TARGET_DATA_PKG;
  with TARGET_EMITTER_ARRAY_PKG; use TARGET_EMITTER_ARRAY_PKG;
-- with/use clauses for generated packages.
  with ATACMS_EXCEPTIONS; use ATACMS_EXCEPTIONS;
  with ATACMS_INSTANTIATIONS; use ATACMS_INSTANTIATIONS;
-- with/use clauses for CAPS library packages.
  with PSDL_STREAMS; use PSDL_STREAMS;
package ATACMS_STREAMS is
-- Local stream instantiations

  package DS_EMISSION_STR_JSTARS_OP is new
    PSDL_STREAMS.SAMPLED_BUFFER(TARGET_EMITTER_ARRAY);

  package DS_FIRE_CMD3_STR_CNR_LINK_OP is new
    PSDL_STREAMS.SAMPLED_BUFFER(TARGET_DATA);

  package DS_FIRE_CMD4_STR_SHOOTER_OP is new
    PSDL_STREAMS.SAMPLED_BUFFER(TARGET_DATA);

  package DS_GUI_OUT_STR_CMDS_OUT is new
    PSDL_STREAMS.SAMPLED_BUFFER(TARGET_DATA);

  package DS_TARGET_ARRAY1_STR_SCDL_LINK_OP is new
    PSDL_STREAMS.SAMPLED_BUFFER(JSTARS_ARRAY);

  package DS_TARGET_ARRAY2_STR_GRND_STAT_MOD_OP is new
    PSDL_STREAMS.SAMPLED_BUFFER(JSTARS_ARRAY);

  package DS_FIRE_CMD1_STR_LAN2_LINK_OP is new
    PSDL_STREAMS.SAMPLED_BUFFER(TARGET_DATA);

  package DS_FIRE_CMD2_STR_CTOC_OP is new
    PSDL_STREAMS.SAMPLED_BUFFER(TARGET_DATA);

  package DS_TARGET_ARRAY3_STR_LAN1_LINK_OP is new
    PSDL_STREAMS.SAMPLED_BUFFER(GRND_STAT_MOD_ARRAY);

  package DS_TARGET_ARRAY4_STR_ASAS_OP is new
    PSDL_STREAMS.SAMPLED_BUFFER(GRND_STAT_MOD_ARRAY);

-- State stream instantiations
```

```
  package DS_GUI_IN_STR_ASAS_OP is new
    PSDL_STREAMS.STATE_VARIABLE(MY_UNIT_PKG.MY_UNIT, PAUSE);

  package DS_GUI_IN_STR_JSTARS_OP is new
    PSDL_STREAMS.STATE_VARIABLE(MY_UNIT_PKG.MY_UNIT, PAUSE);

  package DS_GUI_IN_STR_TARGET_EMITTER_OP is new
    PSDL_STREAMS.STATE_VARIABLE(MY_UNIT_PKG.MY_UNIT, PAUSE);

 end ATACMS_STREAMS;

 package ATACMS_DRIVERS is
  procedure CNR_LINK_OP_DRIVER;
  procedure JSTARS_OP_DRIVER;
  procedure SCDL_LINK_OP_DRIVER;
  procedure SHOOTER_OP_DRIVER;
  procedure TARGET_EMITTER_OP_DRIVER;
  procedure ASAS_OP_DRIVER;
  procedure CTOC_OP_DRIVER;
  procedure GRND_STAT_MOD_OP_DRIVER;
  procedure LAN1_LINK_OP_DRIVER;
  procedure LAN2_LINK_OP_DRIVER;
  procedure CHOOSE_INPUTS_DRIVER;
  procedure GUI_INPUT_EVENT_MONITOR_DRIVER;
  procedure CMDS_OUT_DRIVER;
 end ATACMS_DRIVERS;

-- with/use clauses for atomic components.
  with GRND_STAT_MOD_ARRAY_PKG; use GRND_STAT_MOD_ARRAY_PKG;
  with JSTARS_ARRAY_PKG; use JSTARS_ARRAY_PKG;
  with MY_UNIT_PKG; use MY_UNIT_PKG;
  with TARGET_DATA_PKG; use TARGET_DATA_PKG;
  with TARGET_EMITTER_ARRAY_PKG; use TARGET_EMITTER_ARRAY_PKG;
  with ASAS_OP_PKG; use ASAS_OP_PKG;
  with CHOOSE_INPUTS_PKG; use CHOOSE_INPUTS_PKG;
  with CMDS_OUT_PKG; use CMDS_OUT_PKG;
  with CNR_LINK_OP_PKG; use CNR_LINK_OP_PKG;
  with CTOC_OP_PKG; use CTOC_OP_PKG;
  with GRND_STAT_MOD_OP_PKG; use GRND_STAT_MOD_OP_PKG;
  with GUI_INPUT_EVENT_MONITOR_PKG; use GUI_INPUT_EVENT_MONITOR_PKG;
  with JSTARS_OP_PKG; use JSTARS_OP_PKG;
  with LAN1_LINK_OP_PKG; use LAN1_LINK_OP_PKG;
  with LAN2_LINK_OP_PKG; use LAN2_LINK_OP_PKG;
  with SCDL_LINK_OP_PKG; use SCDL_LINK_OP_PKG;
  with SHOOTER_OP_PKG; use SHOOTER_OP_PKG;
  with TARGET_EMITTER_OP_PKG; use TARGET_EMITTER_OP_PKG;
-- with/use clauses for generated packages.
  with ATACMS_EXCEPTIONS; use ATACMS_EXCEPTIONS;
  with ATACMS_STREAMS; use ATACMS_STREAMS;
  with ATACMS_TIMERS; use ATACMS_TIMERS;
  with ATACMS_INSTANTIATIONS; use ATACMS_INSTANTIATIONS;
-- with/use clauses for CAPS library packages.
  with DS_DEBUG_PKG; use DS_DEBUG_PKG;
  with PSDL_STREAMS; use PSDL_STREAMS;
  with PSDL_TIMERS;
 package body ATACMS_DRIVERS is

  procedure CNR_LINK_OP_DRIVER is
   LV_FIRE_CMD3_STR : TARGET_DATA_PKG.TARGET_DATA;
   LV_FIRE_CMD4_STR : TARGET_DATA_PKG.TARGET_DATA;

   EXCEPTION_HAS_OCCURRED: BOOLEAN := FALSE;
   EXCEPTION_ID: PSDL_EXCEPTION;
  begin
-- Data trigger checks.
   if not (DS_FIRE_CMD3_STR_CNR_LINK_OP.NEW_DATA) then
    return;
   end if;
```

```
-- Data stream reads.
   begin
    DS_FIRE_CMD3_STR_CNR_LINK_OP.BUFFER.READ(LV_FIRE_CMD3_STR);
   exception
    when BUFFER_UNDERFLOW =>
     DS_DEBUG.BUFFER_UNDERFLOW("FIRE_CMD3_STR_CNR_LINK_OP", "CNR_LINK_OP");
   end;

-- Execution trigger condition check.
   if True then
    begin
    CNR_LINK_OP(
     FIRE_CMD3_STR => LV_FIRE_CMD3_STR,
     FIRE_CMD4_STR => LV_FIRE_CMD4_STR);
    exception
     when others =>
      DS_DEBUG.UNDECLARED_EXCEPTION("CNR_LINK_OP");
      EXCEPTION_HAS_OCCURRED := true;
      EXCEPTION_ID := UNDECLARED_ADA_EXCEPTION;
    end;
   else return;
   end if;

-- Exception Constraint translations.

-- Other constraint option translations.

--Unconditional output translations.
   if not EXCEPTION_HAS_OCCURRED then
    begin
     DS_FIRE_CMD4_STR_SHOOTER_OP.BUFFER.WRITE(LV_FIRE_CMD4_STR);
    exception
     when BUFFER_OVERFLOW =>
      DS_DEBUG.BUFFER_OVERFLOW("FIRE_CMD4_STR_SHOOTER_OP", "CNR_LINK_OP");
    end;
   end if;

-- PSDL Exception handler.
   if EXCEPTION_HAS_OCCURRED then
    DS_DEBUG.UNHANDLED_EXCEPTION(
     "CNR_LINK_OP",
     PSDL_EXCEPTION'IMAGE(EXCEPTION_ID));
   end if;
  end CNR_LINK_OP_DRIVER;


  procedure JSTARS_OP_DRIVER is
   LV_EMISSION_STR : TARGET_EMITTER_ARRAY_PKG.TARGET_EMITTER_ARRAY;
   LV_GUI_IN_STR : MY_UNIT_PKG.MY_UNIT;
   LV_TARGET_ARRAY1_STR : JSTARS_ARRAY_PKG.JSTARS_ARRAY;

   EXCEPTION_HAS_OCCURRED: BOOLEAN := FALSE;
   EXCEPTION_ID: PSDL_EXCEPTION;
  begin
-- Data trigger checks.

-- Data stream reads.
   begin
    DS_EMISSION_STR_JSTARS_OP.BUFFER.READ(LV_EMISSION_STR);
   exception
    when BUFFER_UNDERFLOW =>
     DS_DEBUG.BUFFER_UNDERFLOW("EMISSION_STR_JSTARS_OP", "JSTARS_OP");
   end;
   begin
    DS_GUI_IN_STR_JSTARS_OP.BUFFER.READ(LV_GUI_IN_STR);
   exception
    when BUFFER_UNDERFLOW =>
     DS_DEBUG.BUFFER_UNDERFLOW("GUI_IN_STR_JSTARS_OP", "JSTARS_OP");
   end;
```

```
-- Execution trigger condition check.
   if (LV_GUI_IN_STR /= MY_UNIT_PKG.PAUSE) then
    begin
    JSTARS_OP(
      EMISSION_STR => LV_EMISSION_STR,
      GUI_IN_STR => LV_GUI_IN_STR,
      TARGET_ARRAY1_STR => LV_TARGET_ARRAY1_STR);
    exception
      when others =>
        DS_DEBUG.UNDECLARED_EXCEPTION("JSTARS_OP");
        EXCEPTION_HAS_OCCURRED := true;
        EXCEPTION_ID := UNDECLARED_ADA_EXCEPTION;
    end;
   else return;
   end if;

-- Exception Constraint translations.

-- Other constraint option translations.

--Unconditional output translations.
   if not EXCEPTION_HAS_OCCURRED then
    begin
      DS_TARGET_ARRAY1_STR_SCDL_LINK_OP.BUFFER.WRITE(LV_TARGET_ARRAY1_STR);
    exception
      when BUFFER_OVERFLOW =>
        DS_DEBUG.BUFFER_OVERFLOW("TARGET_ARRAY1_STR_SCDL_LINK_OP", "JSTARS_OP");
    end;
   end if;

-- PSDL Exception handler.
   if EXCEPTION_HAS_OCCURRED then
    DS_DEBUG.UNHANDLED_EXCEPTION(
      "JSTARS_OP",
      PSDL_EXCEPTION'IMAGE(EXCEPTION_ID));
   end if;
  end JSTARS_OP_DRIVER;


  procedure SCDL_LINK_OP_DRIVER is
   LV_TARGET_ARRAY1_STR : JSTARS_ARRAY_PKG.JSTARS_ARRAY;
   LV_TARGET_ARRAY2_STR : JSTARS_ARRAY_PKG.JSTARS_ARRAY;

   EXCEPTION_HAS_OCCURRED: BOOLEAN := FALSE;
   EXCEPTION_ID: PSDL_EXCEPTION;
  begin
-- Data trigger checks.
   if not (DS_TARGET_ARRAY1_STR_SCDL_LINK_OP.NEW_DATA) then
    return;
   end if;

-- Data stream reads.
   begin
    DS_TARGET_ARRAY1_STR_SCDL_LINK_OP.BUFFER.READ(LV_TARGET_ARRAY1_STR);
   exception
    when BUFFER_UNDERFLOW =>
      DS_DEBUG.BUFFER_UNDERFLOW("TARGET_ARRAY1_STR_SCDL_LINK_OP", "SCDL_LINK_OP");
   end;

-- Execution trigger condition check.
   if True then
    begin
    SCDL_LINK_OP(
      TARGET_ARRAY1_STR => LV_TARGET_ARRAY1_STR,
      TARGET_ARRAY2_STR => LV_TARGET_ARRAY2_STR);
```

```
        exception
        when others =>
         DS_DEBUG.UNDECLARED_EXCEPTION("SCDL_LINK_OP");
         EXCEPTION_HAS_OCCURRED := true;
         EXCEPTION_ID := UNDECLARED_ADA_EXCEPTION;
      end;
    else return;
    end if;

  -- Exception Constraint translations.

  -- Other constraint option translations.

  --Unconditional output translations.
    if not EXCEPTION_HAS_OCCURRED then
     begin
      DS_TARGET_ARRAY2_STR_GRND_STAT_MOD_OP.BUFFER.WRITE(LV_TARGET_ARRAY2_STR);
     exception
      when BUFFER_OVERFLOW =>
       DS_DEBUG.BUFFER_OVERFLOW("TARGET_ARRAY2_STR_GRND_STAT_MOD_OP", "SCDL_LINK_OP");
     end;
    end if;

  -- PSDL Exception handler.
    if EXCEPTION_HAS_OCCURRED then
     DS_DEBUG.UNHANDLED_EXCEPTION(
      "SCDL_LINK_OP",
      PSDL_EXCEPTION'IMAGE(EXCEPTION_ID));
    end if;
   end SCDL_LINK_OP_DRIVER;


   procedure SHOOTER_OP_DRIVER is
    LV_FIRE_CMD4_STR : TARGET_DATA_PKG.TARGET_DATA;
    LV_GUI_OUT_STR : TARGET_DATA_PKG.TARGET_DATA;

    EXCEPTION_HAS_OCCURRED: BOOLEAN := FALSE;
    EXCEPTION_ID: PSDL_EXCEPTION;
   begin
 -- Data trigger checks.
    if not (DS_FIRE_CMD4_STR_SHOOTER_OP.NEW_DATA) then
     return;
    end if;

 -- Data stream reads.
    begin
     DS_FIRE_CMD4_STR_SHOOTER_OP.BUFFER.READ(LV_FIRE_CMD4_STR);
    exception
     when BUFFER_UNDERFLOW =>
      DS_DEBUG.BUFFER_UNDERFLOW("FIRE_CMD4_STR_SHOOTER_OP", "SHOOTER_OP");
    end;

 -- Execution trigger condition check.
    if True then
     begin
     SHOOTER_OP(
      FIRE_CMD4_STR => LV_FIRE_CMD4_STR,
      GUI_OUT_STR => LV_GUI_OUT_STR);
     exception
      when others =>
       DS_DEBUG.UNDECLARED_EXCEPTION("SHOOTER_OP");
       EXCEPTION_HAS_OCCURRED := true;
       EXCEPTION_ID := UNDECLARED_ADA_EXCEPTION;
     end;
   else return;
   end if;
```

```
-- Exception Constraint translations.

-- Other constraint option translations.

--Unconditional output translations.
   if not EXCEPTION_HAS_OCCURRED then
    begin
     DS_GUI_OUT_STR_CMDS_OUT.BUFFER.WRITE(LV_GUI_OUT_STR);
    exception
     when BUFFER_OVERFLOW =>
      DS_DEBUG.BUFFER_OVERFLOW("GUI_OUT_STR_CMDS_OUT", "SHOOTER_OP");
    end;
   end if;

-- PSDL Exception handler.
   if EXCEPTION_HAS_OCCURRED then
    DS_DEBUG.UNHANDLED_EXCEPTION(
     "SHOOTER_OP",
     PSDL_EXCEPTION'IMAGE(EXCEPTION_ID));
   end if;
  end SHOOTER_OP_DRIVER;


  procedure TARGET_EMITTER_OP_DRIVER is
   LV_GUI_IN_STR : MY_UNIT_PKG.MY_UNIT;
   LV_EMISSION_STR : TARGET_EMITTER_ARRAY_PKG.TARGET_EMITTER_ARRAY;

   EXCEPTION_HAS_OCCURRED: BOOLEAN := FALSE;
   EXCEPTION_ID: PSDL_EXCEPTION;
  begin
-- Data trigger checks.

-- Data stream reads.
   begin
    DS_GUI_IN_STR_TARGET_EMITTER_OP.BUFFER.READ(LV_GUI_IN_STR);
   exception
    when BUFFER_UNDERFLOW =>
     DS_DEBUG.BUFFER_UNDERFLOW("GUI_IN_STR_TARGET_EMITTER_OP", "TARGET_EMITTER_OP");
   end;

-- Execution trigger condition check.
   if (LV_GUI_IN_STR /= MY_UNIT_PKG.PAUSE)  then
    begin
    TARGET_EMITTER_OP(
     GUI_IN_STR => LV_GUI_IN_STR,
     EMISSION_STR => LV_EMISSION_STR);
    exception
     when others =>
      DS_DEBUG.UNDECLARED_EXCEPTION("TARGET_EMITTER_OP");
      EXCEPTION_HAS_OCCURRED := true;
      EXCEPTION_ID := UNDECLARED_ADA_EXCEPTION;
    end;
   else return;
   end if;

-- Exception Constraint translations.

-- Other constraint option translations.

--Unconditional output translations.
   if not EXCEPTION_HAS_OCCURRED then
    begin
     DS_EMISSION_STR_JSTARS_OP.BUFFER.WRITE(LV_EMISSION_STR);
    exception
     when BUFFER_OVERFLOW =>
      DS_DEBUG.BUFFER_OVERFLOW("EMISSION_STR_JSTARS_OP", "TARGET_EMITTER_OP");
    end;
   end if;
```

```
-- PSDL Exception handler.
   if EXCEPTION_HAS_OCCURRED then
     DS_DEBUG.UNHANDLED_EXCEPTION(
       "TARGET_EMITTER_OP",
       PSDL_EXCEPTION'IMAGE(EXCEPTION_ID));
   end if;
 end TARGET_EMITTER_OP_DRIVER;


 procedure ASAS_OP_DRIVER is
  LV_GUI_IN_STR : MY_UNIT_PKG.MY_UNIT;
  LV_TARGET_ARRAY4_STR : GRND_STAT_MOD_ARRAY_PKG.GRND_STAT_MOD_ARRAY;
  LV_FIRE_CMD1_STR : TARGET_DATA_PKG.TARGET_DATA;

  EXCEPTION_HAS_OCCURRED: BOOLEAN := FALSE;
  EXCEPTION_ID: PSDL_EXCEPTION;
 begin
-- Data trigger checks.

-- Data stream reads.
   begin
     DS_GUI_IN_STR_ASAS_OP.BUFFER.READ(LV_GUI_IN_STR);
   exception
     when BUFFER_UNDERFLOW =>
       DS_DEBUG.BUFFER_UNDERFLOW("GUI_IN_STR_ASAS_OP", "ASAS_OP");
   end;
   begin
     DS_TARGET_ARRAY4_STR_ASAS_OP.BUFFER.READ(LV_TARGET_ARRAY4_STR);
   exception
     when BUFFER_UNDERFLOW =>
       DS_DEBUG.BUFFER_UNDERFLOW("TARGET_ARRAY4_STR_ASAS_OP", "ASAS_OP");
   end;

-- Execution trigger condition check.
   if (LV_GUI_IN_STR /= MY_UNIT_PKG.PAUSE) then
     begin
     ASAS_OP(
       GUI_IN_STR => LV_GUI_IN_STR,
       TARGET_ARRAY4_STR => LV_TARGET_ARRAY4_STR,
       FIRE_CMD1_STR => LV_FIRE_CMD1_STR);
     exception
       when others =>
         DS_DEBUG.UNDECLARED_EXCEPTION("ASAS_OP");
         EXCEPTION_HAS_OCCURRED := true;
         EXCEPTION_ID := UNDECLARED_ADA_EXCEPTION;
     end;
   else return;
   end if;

-- Exception Constraint translations.

-- Other constraint option translations.

--Unconditional output translations.
   if not EXCEPTION_HAS_OCCURRED then
     begin
       DS_FIRE_CMD1_STR_LAN2_LINK_OP.BUFFER.WRITE(LV_FIRE_CMD1_STR);
     exception
       when BUFFER_OVERFLOW =>
         DS_DEBUG.BUFFER_OVERFLOW("FIRE_CMD1_STR_LAN2_LINK_OP", "ASAS_OP");
     end;
   end if;

-- PSDL Exception handler.
   if EXCEPTION_HAS_OCCURRED then
     DS_DEBUG.UNHANDLED_EXCEPTION(
       "ASAS_OP",
       PSDL_EXCEPTION'IMAGE(EXCEPTION_ID));
   end if;
```

```
end ASAS_OP_DRIVER;


procedure CTOC_OP_DRIVER is
 LV_FIRE_CMD2_STR : TARGET_DATA_PKG.TARGET_DATA;
 LV_FIRE_CMD3_STR : TARGET_DATA_PKG.TARGET_DATA;

 EXCEPTION_HAS_OCCURRED: BOOLEAN := FALSE;
 EXCEPTION_ID: PSDL_EXCEPTION;
begin
-- Data trigger checks.
 if not (DS_FIRE_CMD2_STR_CTOC_OP.NEW_DATA) then
  return;
 end if;

-- Data stream reads.
 begin
  DS_FIRE_CMD2_STR_CTOC_OP.BUFFER.READ(LV_FIRE_CMD2_STR);
 exception
  when BUFFER_UNDERFLOW =>
   DS_DEBUG.BUFFER_UNDERFLOW("FIRE_CMD2_STR_CTOC_OP", "CTOC_OP");
 end;

-- Execution trigger condition check.
 if True then
  begin
  CTOC_OP(
   FIRE_CMD2_STR => LV_FIRE_CMD2_STR,
   FIRE_CMD3_STR => LV_FIRE_CMD3_STR);
  exception
   when others =>
    DS_DEBUG.UNDECLARED_EXCEPTION("CTOC_OP");
    EXCEPTION_HAS_OCCURRED := true;
    EXCEPTION_ID := UNDECLARED_ADA_EXCEPTION;
  end;
 else return;
 end if;

-- Exception Constraint translations.

-- Other constraint option translations.

--Unconditional output translations.
 if not EXCEPTION_HAS_OCCURRED then
  begin
   DS_FIRE_CMD3_STR_CNR_LINK_OP.BUFFER.WRITE(LV_FIRE_CMD3_STR);
  exception
   when BUFFER_OVERFLOW =>
    DS_DEBUG.BUFFER_OVERFLOW("FIRE_CMD3_STR_CNR_LINK_OP", "CTOC_OP");
  end;
 end if;

-- PSDL Exception handler.
 if EXCEPTION_HAS_OCCURRED then
  DS_DEBUG.UNHANDLED_EXCEPTION(
   "CTOC_OP",
   PSDL_EXCEPTION'IMAGE(EXCEPTION_ID));
 end if;
end CTOC_OP_DRIVER;


procedure GRND_STAT_MOD_OP_DRIVER is
 LV_TARGET_ARRAY2_STR : JSTARS_ARRAY_PKG.JSTARS_ARRAY;
 LV_TARGET_ARRAY3_STR : GRND_STAT_MOD_ARRAY_PKG.GRND_STAT_MOD_ARRAY;

 EXCEPTION_HAS_OCCURRED: BOOLEAN := FALSE;
 EXCEPTION_ID: PSDL_EXCEPTION;
begin
```

```
-- Data trigger checks.
   if not (DS_TARGET_ARRAY2_STR_GRND_STAT_MOD_OP.NEW_DATA) then
    return;
   end if;

-- Data stream reads.
   begin
    DS_TARGET_ARRAY2_STR_GRND_STAT_MOD_OP.BUFFER.READ(LV_TARGET_ARRAY2_STR);
   exception
    when BUFFER_UNDERFLOW =>
     DS_DEBUG.BUFFER_UNDERFLOW("TARGET_ARRAY2_STR_GRND_STAT_MOD_OP", "GRND_STAT_MOD_OP");
   end;

-- Execution trigger condition check.
   if True then
    begin
    GRND_STAT_MOD_OP(
     TARGET_ARRAY2_STR => LV_TARGET_ARRAY2_STR,
     TARGET_ARRAY3_STR => LV_TARGET_ARRAY3_STR);
    exception
     when others =>
      DS_DEBUG.UNDECLARED_EXCEPTION("GRND_STAT_MOD_OP");
      EXCEPTION_HAS_OCCURRED := true;
      EXCEPTION_ID := UNDECLARED_ADA_EXCEPTION;
    end;
   else return;
   end if;

-- Exception Constraint translations.

-- Other constraint option translations.

--Unconditional output translations.
   if not EXCEPTION_HAS_OCCURRED then
    begin
     DS_TARGET_ARRAY3_STR_LAN1_LINK_OP.BUFFER.WRITE(LV_TARGET_ARRAY3_STR);
    exception
     when BUFFER_OVERFLOW =>
      DS_DEBUG.BUFFER_OVERFLOW("TARGET_ARRAY3_STR_LAN1_LINK_OP", "GRND_STAT_MOD_OP");
    end;
   end if;

-- PSDL Exception handler.
   if EXCEPTION_HAS_OCCURRED then
    DS_DEBUG.UNHANDLED_EXCEPTION(
     "GRND_STAT_MOD_OP",
     PSDL_EXCEPTION'IMAGE(EXCEPTION_ID));
   end if;
  end GRND_STAT_MOD_OP_DRIVER;


  procedure LAN1_LINK_OP_DRIVER is
   LV_TARGET_ARRAY3_STR : GRND_STAT_MOD_ARRAY_PKG.GRND_STAT_MOD_ARRAY;
   LV_TARGET_ARRAY4_STR : GRND_STAT_MOD_ARRAY_PKG.GRND_STAT_MOD_ARRAY;

   EXCEPTION_HAS_OCCURRED: BOOLEAN := FALSE;
   EXCEPTION_ID: PSDL_EXCEPTION;
  begin
-- Data trigger checks.
   if not (DS_TARGET_ARRAY3_STR_LAN1_LINK_OP.NEW_DATA) then
    return;
   end if;
```

```
-- Data stream reads.
   begin
    DS_TARGET_ARRAY3_STR_LAN1_LINK_OP.BUFFER.READ(LV_TARGET_ARRAY3_STR);
   exception
    when BUFFER_UNDERFLOW =>
     DS_DEBUG.BUFFER_UNDERFLOW("TARGET_ARRAY3_STR_LAN1_LINK_OP", "LAN1_LINK_OP");
   end;

-- Execution trigger condition check.
   if True then
    begin
    LAN1_LINK_OP(
     TARGET_ARRAY3_STR => LV_TARGET_ARRAY3_STR,
     TARGET_ARRAY4_STR => LV_TARGET_ARRAY4_STR);
    exception
     when others =>
      DS_DEBUG.UNDECLARED_EXCEPTION("LAN1_LINK_OP");
      EXCEPTION_HAS_OCCURRED := true;
      EXCEPTION_ID := UNDECLARED_ADA_EXCEPTION;
    end;
   else return;
   end if;

-- Exception Constraint translations.

-- Other constraint option translations.

--Unconditional output translations.
   if not EXCEPTION_HAS_OCCURRED then
    begin
     DS_TARGET_ARRAY4_STR_ASAS_OP.BUFFER.WRITE(LV_TARGET_ARRAY4_STR);
    exception
     when BUFFER_OVERFLOW =>
      DS_DEBUG.BUFFER_OVERFLOW("TARGET_ARRAY4_STR_ASAS_OP", "LAN1_LINK_OP");
    end;
   end if;

-- PSDL Exception handler.
   if EXCEPTION_HAS_OCCURRED then
    DS_DEBUG.UNHANDLED_EXCEPTION(
     "LAN1_LINK_OP",
     PSDL_EXCEPTION'IMAGE(EXCEPTION_ID));
   end if;
  end LAN1_LINK_OP_DRIVER;


  procedure LAN2_LINK_OP_DRIVER is
   LV_FIRE_CMD1_STR : TARGET_DATA_PKG.TARGET_DATA;
   LV_FIRE_CMD2_STR : TARGET_DATA_PKG.TARGET_DATA;

   EXCEPTION_HAS_OCCURRED: BOOLEAN := FALSE;
   EXCEPTION_ID: PSDL_EXCEPTION;
  begin
-- Data trigger checks.
   if not (DS_FIRE_CMD1_STR_LAN2_LINK_OP.NEW_DATA) then
    return;
   end if;

-- Data stream reads.
   begin
    DS_FIRE_CMD1_STR_LAN2_LINK_OP.BUFFER.READ(LV_FIRE_CMD1_STR);
   exception
    when BUFFER_UNDERFLOW =>
     DS_DEBUG.BUFFER_UNDERFLOW("FIRE_CMD1_STR_LAN2_LINK_OP", "LAN2_LINK_OP");
   end;
```

```
-- Execution trigger condition check.
   if True then
    begin
    LAN2_LINK_OP(
     FIRE_CMD1_STR => LV_FIRE_CMD1_STR,
     FIRE_CMD2_STR => LV_FIRE_CMD2_STR);
    exception
     when others =>
      DS_DEBUG.UNDECLARED_EXCEPTION("LAN2_LINK_OP");
      EXCEPTION_HAS_OCCURRED := true;
      EXCEPTION_ID := UNDECLARED_ADA_EXCEPTION;
    end;
   else return;
   end if;

-- Exception Constraint translations.

-- Other constraint option translations.

--Unconditional output translations.
   if not EXCEPTION_HAS_OCCURRED then
    begin
     DS_FIRE_CMD2_STR_CTOC_OP.BUFFER.WRITE(LV_FIRE_CMD2_STR);
    exception
     when BUFFER_OVERFLOW =>
      DS_DEBUG.BUFFER_OVERFLOW("FIRE_CMD2_STR_CTOC_OP", "LAN2_LINK_OP");
    end;
   end if;

-- PSDL Exception handler.
   if EXCEPTION_HAS_OCCURRED then
    DS_DEBUG.UNHANDLED_EXCEPTION(
     "LAN2_LINK_OP",
     PSDL_EXCEPTION'IMAGE(EXCEPTION_ID));
   end if;
  end LAN2_LINK_OP_DRIVER;


  procedure CHOOSE_INPUTS_DRIVER is
   LV_GUI_IN_STR : MY_UNIT_PKG.MY_UNIT;

   EXCEPTION_HAS_OCCURRED: BOOLEAN := FALSE;
   EXCEPTION_ID: PSDL_EXCEPTION;
  begin
-- Data trigger checks.

-- Data stream reads.

-- Execution trigger condition check.
   if True then
    begin
    CHOOSE_INPUTS(
     GUI_IN_STR => LV_GUI_IN_STR);
    exception
     when others =>
      DS_DEBUG.UNDECLARED_EXCEPTION("CHOOSE_INPUTS");
      EXCEPTION_HAS_OCCURRED := true;
      EXCEPTION_ID := UNDECLARED_ADA_EXCEPTION;
    end;
   else return;
   end if;

-- Exception Constraint translations.

-- Other constraint option translations.
```

```
--Unconditional output translations.
    if not EXCEPTION_HAS_OCCURRED then
      begin
        DS_GUI_IN_STR_ASAS_OP.BUFFER.WRITE(LV_GUI_IN_STR);
      exception
        when BUFFER_OVERFLOW =>
          DS_DEBUG.BUFFER_OVERFLOW("GUI_IN_STR_ASAS_OP", "CHOOSE_INPUTS");
      end;
      begin
        DS_GUI_IN_STR_JSTARS_OP.BUFFER.WRITE(LV_GUI_IN_STR);
      exception
        when BUFFER_OVERFLOW =>
          DS_DEBUG.BUFFER_OVERFLOW("GUI_IN_STR_JSTARS_OP", "CHOOSE_INPUTS");
      end;
      begin
        DS_GUI_IN_STR_TARGET_EMITTER_OP.BUFFER.WRITE(LV_GUI_IN_STR);
      exception
        when BUFFER_OVERFLOW =>
          DS_DEBUG.BUFFER_OVERFLOW("GUI_IN_STR_TARGET_EMITTER_OP", "CHOOSE_INPUTS");
      end;
    end if;

-- PSDL Exception handler.
    if EXCEPTION_HAS_OCCURRED then
      DS_DEBUG.UNHANDLED_EXCEPTION(
        "CHOOSE_INPUTS",
        PSDL_EXCEPTION'IMAGE(EXCEPTION_ID));
    end if;
  end CHOOSE_INPUTS_DRIVER;


  procedure GUI_INPUT_EVENT_MONITOR_DRIVER is

    EXCEPTION_HAS_OCCURRED: BOOLEAN := FALSE;
    EXCEPTION_ID: PSDL_EXCEPTION;
  begin
-- Data trigger checks.

-- Data stream reads.

-- Execution trigger condition check.
    if True then
      begin
      GUI_INPUT_EVENT_MONITOR;
      exception
        when others =>
          DS_DEBUG.UNDECLARED_EXCEPTION("GUI_INPUT_EVENT_MONITOR");
          EXCEPTION_HAS_OCCURRED := true;
          EXCEPTION_ID := UNDECLARED_ADA_EXCEPTION;
      end;
    else return;
    end if;

-- Exception Constraint translations.

-- Other constraint option translations.

--Unconditional output translations.

-- PSDL Exception handler.
    if EXCEPTION_HAS_OCCURRED then
      DS_DEBUG.UNHANDLED_EXCEPTION(
        "GUI_INPUT_EVENT_MONITOR",
        PSDL_EXCEPTION'IMAGE(EXCEPTION_ID));
    end if;
  end GUI_INPUT_EVENT_MONITOR_DRIVER;
```

```
  procedure CMDS_OUT_DRIVER is
   LV_GUI_OUT_STR : TARGET_DATA_PKG.TARGET_DATA;

   EXCEPTION_HAS_OCCURRED: BOOLEAN := FALSE;
   EXCEPTION_ID: PSDL_EXCEPTION;
  begin
-- Data trigger checks.
   if not (DS_GUI_OUT_STR_CMDS_OUT.NEW_DATA) then
    return;
   end if;

-- Data stream reads.
   begin
    DS_GUI_OUT_STR_CMDS_OUT.BUFFER.READ(LV_GUI_OUT_STR);
   exception
    when BUFFER_UNDERFLOW =>
     DS_DEBUG.BUFFER_UNDERFLOW("GUI_OUT_STR_CMDS_OUT", "CMDS_OUT");
   end;

-- Execution trigger condition check.
   if True then
    begin
    CMDS_OUT(
     GUI_OUT_STR => LV_GUI_OUT_STR);
    exception
     when others =>
      DS_DEBUG.UNDECLARED_EXCEPTION("CMDS_OUT");
      EXCEPTION_HAS_OCCURRED := true;
      EXCEPTION_ID := UNDECLARED_ADA_EXCEPTION;
    end;
   else return;
   end if;

-- Exception Constraint translations.

-- Other constraint option translations.

--Unconditional output translations.

-- PSDL Exception handler.
   if EXCEPTION_HAS_OCCURRED then
    DS_DEBUG.UNHANDLED_EXCEPTION(
     "CMDS_OUT",
     PSDL_EXCEPTION'IMAGE(EXCEPTION_ID));
   end if;
  end CMDS_OUT_DRIVER;
end ATACMS_DRIVERS;

package atacms_DYNAMIC_SCHEDULERS is
 procedure START_DYNAMIC_SCHEDULE;
end atacms_DYNAMIC_SCHEDULERS;


with atacms_DRIVERS; use atacms_DRIVERS;
with PRIORITY_DEFINITIONS; use PRIORITY_DEFINITIONS;
package body atacms_DYNAMIC_SCHEDULERS is

 task type DYNAMIC_SCHEDULE_TYPE is
  pragma priority (DYNAMIC_SCHEDULE_PRIORITY);
  entry START;
 end DYNAMIC_SCHEDULE_TYPE;
 for DYNAMIC_SCHEDULE_TYPE'STORAGE_SIZE use 100_000;
 DYNAMIC_SCHEDULE : DYNAMIC_SCHEDULE_TYPE;

 task body DYNAMIC_SCHEDULE_TYPE is
 begin
  accept START;
  loop
```

```
delay 5.0;
    cmds_out_DRIVER;
   end loop;
  end DYNAMIC_SCHEDULE_TYPE;

  procedure START_DYNAMIC_SCHEDULE is
  begin
   DYNAMIC_SCHEDULE.START;
  end START_DYNAMIC_SCHEDULE;

 end atacms_DYNAMIC_SCHEDULERS;


 package atacms_STATIC_SCHEDULERS is
  procedure START_STATIC_SCHEDULE;
 end atacms_STATIC_SCHEDULERS;

 with atacms_DRIVERS;  use atacms_DRIVERS;
 with PRIORITY_DEFINITIONS;  use PRIORITY_DEFINITIONS;
 with PSDL_TIMERS;  use PSDL_TIMERS;
 with TEXT_IO;  use TEXT_IO;
 package body atacms_STATIC_SCHEDULERS is

  task type STATIC_SCHEDULE_TYPE is
   pragma priority (STATIC_SCHEDULE_PRIORITY);
   entry START;
  end STATIC_SCHEDULE_TYPE;
  for STATIC_SCHEDULE_TYPE'STORAGE_SIZE use 200_000;
  STATIC_SCHEDULE : STATIC_SCHEDULE_TYPE;

  task body STATIC_SCHEDULE_TYPE is
   PERIOD : duration;
   target_emitter_op_START_TIME1 : duration;
   target_emitter_op_STOP_TIME1 : duration;
   jstars_op_START_TIME2 : duration;
   jstars_op_STOP_TIME2 : duration;
   scdl_link_op_START_TIME3 : duration;
   scdl_link_op_STOP_TIME3 : duration;
   grnd_stat_mod_op_START_TIME4 : duration;
   grnd_stat_mod_op_STOP_TIME4 : duration;
   lan1_link_op_START_TIME5 : duration;
   lan1_link_op_STOP_TIME5 : duration;
   asas_op_START_TIME6 : duration;
   asas_op_STOP_TIME6 : duration;
   choose_inputs_START_TIME7 : duration;
   choose_inputs_STOP_TIME7 : duration;
   lan2_link_op_START_TIME8 : duration;
   lan2_link_op_STOP_TIME8 : duration;
   ctoc_op_START_TIME9 : duration;
   ctoc_op_STOP_TIME9 : duration;
   cnr_link_op_START_TIME10 : duration;
   cnr_link_op_STOP_TIME10 : duration;
   shooter_op_START_TIME11 : duration;
   shooter_op_STOP_TIME11 : duration;
   gui_input_event_monitor_START_TIME12 : duration;
   gui_input_event_monitor_STOP_TIME12 : duration;
   scdl_link_op_START_TIME13 : duration;
   scdl_link_op_STOP_TIME13 : duration;
   lan1_link_op_START_TIME14 : duration;
   lan1_link_op_STOP_TIME14 : duration;
   choose_inputs_START_TIME15 : duration;
   choose_inputs_STOP_TIME15 : duration;
   lan2_link_op_START_TIME16 : duration;
   lan2_link_op_STOP_TIME16 : duration;
   ctoc_op_START_TIME17 : duration;
   ctoc_op_STOP_TIME17 : duration;
   cnr_link_op_START_TIME18 : duration;
   cnr_link_op_STOP_TIME18 : duration;
   shooter_op_START_TIME19 : duration;
```

```
shooter_op_STOP_TIME19 : duration;
scdl_link_op_START_TIME20 : duration;
scdl_link_op_STOP_TIME20 : duration;
grnd_stat_mod_op_START_TIME21 : duration;
grnd_stat_mod_op_STOP_TIME21 : duration;
lan1_link_op_START_TIME22 : duration;
lan1_link_op_STOP_TIME22 : duration;
asas_op_START_TIME23 : duration;
asas_op_STOP_TIME23 : duration;
choose_inputs_START_TIME24 : duration;
choose_inputs_STOP_TIME24 : duration;
lan2_link_op_START_TIME25 : duration;
lan2_link_op_STOP_TIME25 : duration;
ctoc_op_START_TIME26 : duration;
ctoc_op_STOP_TIME26 : duration;
cnr_link_op_START_TIME27 : duration;
cnr_link_op_STOP_TIME27 : duration;
shooter_op_START_TIME28 : duration;
shooter_op_STOP_TIME28 : duration;
gui_input_event_monitor_START_TIME29 : duration;
gui_input_event_monitor_STOP_TIME29 : duration;
scdl_link_op_START_TIME30 : duration;
scdl_link_op_STOP_TIME30 : duration;
lan1_link_op_START_TIME31 : duration;
lan1_link_op_STOP_TIME31 : duration;
choose_inputs_START_TIME32 : duration;
choose_inputs_STOP_TIME32 : duration;
lan2_link_op_START_TIME33 : duration;
lan2_link_op_STOP_TIME33 : duration;
ctoc_op_START_TIME34 : duration;
ctoc_op_STOP_TIME34 : duration;
cnr_link_op_START_TIME35 : duration;
cnr_link_op_STOP_TIME35 : duration;
shooter_op_START_TIME36 : duration;
shooter_op_STOP_TIME36 : duration;
jstars_op_START_TIME37 : duration;
jstars_op_STOP_TIME37 : duration;
scdl_link_op_START_TIME38 : duration;
scdl_link_op_STOP_TIME38 : duration;
grnd_stat_mod_op_START_TIME39 : duration;
grnd_stat_mod_op_STOP_TIME39 : duration;
lan1_link_op_START_TIME40 : duration;
lan1_link_op_STOP_TIME40 : duration;
asas_op_START_TIME41 : duration;
asas_op_STOP_TIME41 : duration;
choose_inputs_START_TIME42 : duration;
choose_inputs_STOP_TIME42 : duration;
lan2_link_op_START_TIME43 : duration;
lan2_link_op_STOP_TIME43 : duration;
ctoc_op_START_TIME44 : duration;
ctoc_op_STOP_TIME44 : duration;
cnr_link_op_START_TIME45 : duration;
cnr_link_op_STOP_TIME45 : duration;
shooter_op_START_TIME46 : duration;
shooter_op_STOP_TIME46 : duration;
gui_input_event_monitor_START_TIME47 : duration;
gui_input_event_monitor_STOP_TIME47 : duration;
scdl_link_op_START_TIME48 : duration;
scdl_link_op_STOP_TIME48 : duration;
lan1_link_op_START_TIME49 : duration;
lan1_link_op_STOP_TIME49 : duration;
choose_inputs_START_TIME50 : duration;
choose_inputs_STOP_TIME50 : duration;
lan2_link_op_START_TIME51 : duration;
lan2_link_op_STOP_TIME51 : duration;
ctoc_op_START_TIME52 : duration;
ctoc_op_STOP_TIME52 : duration;
cnr_link_op_START_TIME53 : duration;
cnr_link_op_STOP_TIME53 : duration;
```

```
 shooter_op_START_TIME54 : duration;
 shooter_op_STOP_TIME54 : duration;
 scdl_link_op_START_TIME55 : duration;
 scdl_link_op_STOP_TIME55 : duration;
 grnd_stat_mod_op_START_TIME56 : duration;
 grnd_stat_mod_op_STOP_TIME56 : duration;
 lan1_link_op_START_TIME57 : duration;
 lan1_link_op_STOP_TIME57 : duration;
 asas_op_START_TIME58 : duration;
 asas_op_STOP_TIME58 : duration;
 choose_inputs_START_TIME59 : duration;
 choose_inputs_STOP_TIME59 : duration;
 lan2_link_op_START_TIME60 : duration;
 lan2_link_op_STOP_TIME60 : duration;
 ctoc_op_START_TIME61 : duration;
 ctoc_op_STOP_TIME61 : duration;
 cnr_link_op_START_TIME62 : duration;
 cnr_link_op_STOP_TIME62 : duration;
 shooter_op_START_TIME63 : duration;
 shooter_op_STOP_TIME63 : duration;
 gui_input_event_monitor_START_TIME64 : duration;
 gui_input_event_monitor_STOP_TIME64 : duration;
 scdl_link_op_START_TIME65 : duration;
 scdl_link_op_STOP_TIME65 : duration;
 lan1_link_op_START_TIME66 : duration;
 lan1_link_op_STOP_TIME66 : duration;
 choose_inputs_START_TIME67 : duration;
 choose_inputs_STOP_TIME67 : duration;
 lan2_link_op_START_TIME68 : duration;
 lan2_link_op_STOP_TIME68 : duration;
 ctoc_op_START_TIME69 : duration;
 ctoc_op_STOP_TIME69 : duration;
 cnr_link_op_START_TIME70 : duration;
 cnr_link_op_STOP_TIME70 : duration;
 shooter_op_START_TIME71 : duration;
 shooter_op_STOP_TIME71 : duration;
 schedule_timer : TIMER := NEW_TIMER;
begin
 accept START;
 PERIOD := TARGET_TO_HOST(duration( 1.60000000000000E+01));
 target_emitter_op_START_TIME1 := TARGET_TO_HOST(duration( 0.00000000000000E+00));
 target_emitter_op_STOP_TIME1 := TARGET_TO_HOST(duration( 5.00000000000000E-01));
 jstars_op_START_TIME2 := TARGET_TO_HOST(duration( 5.00000000000000E-01));
 jstars_op_STOP_TIME2 := TARGET_TO_HOST(duration( 1.00000000000000E+00));
 scdl_link_op_START_TIME3 := TARGET_TO_HOST(duration( 1.00000000000000E+00));
 scdl_link_op_STOP_TIME3 := TARGET_TO_HOST(duration( 1.05000000000000E+00));
 grnd_stat_mod_op_START_TIME4 := TARGET_TO_HOST(duration( 1.05000000000000E+00));
 grnd_stat_mod_op_STOP_TIME4 := TARGET_TO_HOST(duration( 1.10000000000000E+00));
 lan1_link_op_START_TIME5 := TARGET_TO_HOST(duration( 1.10000000000000E+00));
 lan1_link_op_STOP_TIME5 := TARGET_TO_HOST(duration( 1.15000000000000E+00));
 asas_op_START_TIME6 := TARGET_TO_HOST(duration( 1.15000000000000E+00));
 asas_op_STOP_TIME6 := TARGET_TO_HOST(duration( 1.35000000000000E+00));
 choose_inputs_START_TIME7 := TARGET_TO_HOST(duration( 1.35000000000000E+00));
 choose_inputs_STOP_TIME7 := TARGET_TO_HOST(duration( 1.55000000000000E+00));
 lan2_link_op_START_TIME8 := TARGET_TO_HOST(duration( 1.55000000000000E+00));
 lan2_link_op_STOP_TIME8 := TARGET_TO_HOST(duration( 1.60000000000000E+00));
 ctoc_op_START_TIME9 := TARGET_TO_HOST(duration( 1.60000000000000E+00));
 ctoc_op_STOP_TIME9 := TARGET_TO_HOST(duration( 1.65000000000000E+00));
 cnr_link_op_START_TIME10 := TARGET_TO_HOST(duration( 1.65000000000000E+00));
 cnr_link_op_STOP_TIME10 := TARGET_TO_HOST(duration( 1.70000000000000E+00));
 shooter_op_START_TIME11 := TARGET_TO_HOST(duration( 1.70000000000000E+00));
 shooter_op_STOP_TIME11 := TARGET_TO_HOST(duration( 1.75000000000000E+00));
 gui_input_event_monitor_START_TIME12 := TARGET_TO_HOST(duration( 1.75000000000000E+00));
 gui_input_event_monitor_STOP_TIME12 := TARGET_TO_HOST(duration( 1.95000000000000E+00));
 scdl_link_op_START_TIME13 := TARGET_TO_HOST(duration( 3.00000000000000E+00));
 scdl_link_op_STOP_TIME13 := TARGET_TO_HOST(duration( 3.05000000000000E+00));
 lan1_link_op_START_TIME14 := TARGET_TO_HOST(duration( 3.10000000000000E+00));
 lan1_link_op_STOP_TIME14 := TARGET_TO_HOST(duration( 3.15000000000000E+00));
 choose_inputs_START_TIME15 := TARGET_TO_HOST(duration( 3.35000000000000E+00));
```

```
choose_inputs_STOP_TIME15 := TARGET_TO_HOST(duration( 3.55000000000000E+00));
lan2_link_op_START_TIME16 := TARGET_TO_HOST(duration( 3.55000000000000E+00));
lan2_link_op_STOP_TIME16 := TARGET_TO_HOST(duration( 3.60000000000000E+00));
ctoc_op_START_TIME17 := TARGET_TO_HOST(duration( 3.60000000000000E+00));
ctoc_op_STOP_TIME17 := TARGET_TO_HOST(duration( 3.65000000000000E+00));
cnr_link_op_START_TIME18 := TARGET_TO_HOST(duration( 3.65000000000000E+00));
cnr_link_op_STOP_TIME18 := TARGET_TO_HOST(duration( 3.70000000000000E+00));
shooter_op_START_TIME19 := TARGET_TO_HOST(duration( 3.70000000000000E+00));
shooter_op_STOP_TIME19 := TARGET_TO_HOST(duration( 3.75000000000000E+00));
scdl_link_op_START_TIME20 := TARGET_TO_HOST(duration( 5.00000000000000E+00));
scdl_link_op_STOP_TIME20 := TARGET_TO_HOST(duration( 5.05000000000000E+00));
grnd_stat_mod_op_START_TIME21 := TARGET_TO_HOST(duration( 5.05000000000000E+00));
grnd_stat_mod_op_STOP_TIME21 := TARGET_TO_HOST(duration( 5.10000000000000E+00));
lan1_link_op_START_TIME22 := TARGET_TO_HOST(duration( 5.10000000000000E+00));
lan1_link_op_STOP_TIME22 := TARGET_TO_HOST(duration( 5.15000000000000E+00));
asas_op_START_TIME23 := TARGET_TO_HOST(duration( 5.15000000000000E+00));
asas_op_STOP_TIME23 := TARGET_TO_HOST(duration( 5.35000000000000E+00));
choose_inputs_START_TIME24 := TARGET_TO_HOST(duration( 5.35000000000000E+00));
choose_inputs_STOP_TIME24 := TARGET_TO_HOST(duration( 5.55000000000000E+00));
lan2_link_op_START_TIME25 := TARGET_TO_HOST(duration( 5.55000000000000E+00));
lan2_link_op_STOP_TIME25 := TARGET_TO_HOST(duration( 5.60000000000000E+00));
ctoc_op_START_TIME26 := TARGET_TO_HOST(duration( 5.60000000000000E+00));
ctoc_op_STOP_TIME26 := TARGET_TO_HOST(duration( 5.65000000000000E+00));
cnr_link_op_START_TIME27 := TARGET_TO_HOST(duration( 5.65000000000000E+00));
cnr_link_op_STOP_TIME27 := TARGET_TO_HOST(duration( 5.70000000000000E+00));
shooter_op_START_TIME28 := TARGET_TO_HOST(duration( 5.70000000000000E+00));
shooter_op_STOP_TIME28 := TARGET_TO_HOST(duration( 5.75000000000000E+00));
gui_input_event_monitor_START_TIME29 := TARGET_TO_HOST(duration( 5.75000000000000E+00));
gui_input_event_monitor_STOP_TIME29 := TARGET_TO_HOST(duration( 5.95000000000000E+00));
scdl_link_op_START_TIME30 := TARGET_TO_HOST(duration( 7.00000000000000E+00));
scdl_link_op_STOP_TIME30 := TARGET_TO_HOST(duration( 7.05000000000000E+00));
lan1_link_op_START_TIME31 := TARGET_TO_HOST(duration( 7.10000000000000E+00));
lan1_link_op_STOP_TIME31 := TARGET_TO_HOST(duration( 7.15000000000000E+00));
choose_inputs_START_TIME32 := TARGET_TO_HOST(duration( 7.35000000000000E+00));
choose_inputs_STOP_TIME32 := TARGET_TO_HOST(duration( 7.55000000000000E+00));
lan2_link_op_START_TIME33 := TARGET_TO_HOST(duration( 7.55000000000000E+00));
lan2_link_op_STOP_TIME33 := TARGET_TO_HOST(duration( 7.60000000000000E+00));
ctoc_op_START_TIME34 := TARGET_TO_HOST(duration( 7.60000000000000E+00));
ctoc_op_STOP_TIME34 := TARGET_TO_HOST(duration( 7.65000000000000E+00));
cnr_link_op_START_TIME35 := TARGET_TO_HOST(duration( 7.65000000000000E+00));
cnr_link_op_STOP_TIME35 := TARGET_TO_HOST(duration( 7.70000000000000E+00));
shooter_op_START_TIME36 := TARGET_TO_HOST(duration( 7.70000000000000E+00));
shooter_op_STOP_TIME36 := TARGET_TO_HOST(duration( 7.75000000000000E+00));
jstars_op_START_TIME37 := TARGET_TO_HOST(duration( 8.50000000000000E+00));
jstars_op_STOP_TIME37 := TARGET_TO_HOST(duration( 9.00000000000000E+00));
scdl_link_op_START_TIME38 := TARGET_TO_HOST(duration( 9.00000000000000E+00));
scdl_link_op_STOP_TIME38 := TARGET_TO_HOST(duration( 9.05000000000000E+00));
grnd_stat_mod_op_START_TIME39 := TARGET_TO_HOST(duration( 9.05000000000000E+00));
grnd_stat_mod_op_STOP_TIME39 := TARGET_TO_HOST(duration( 9.10000000000000E+00));
lan1_link_op_START_TIME40 := TARGET_TO_HOST(duration( 9.10000000000000E+00));
lan1_link_op_STOP_TIME40 := TARGET_TO_HOST(duration( 9.15000000000000E+00));
asas_op_START_TIME41 := TARGET_TO_HOST(duration( 9.15000000000000E+00));
asas_op_STOP_TIME41 := TARGET_TO_HOST(duration( 9.35000000000000E+00));
choose_inputs_START_TIME42 := TARGET_TO_HOST(duration( 9.35000000000000E+00));
choose_inputs_STOP_TIME42 := TARGET_TO_HOST(duration( 9.55000000000000E+00));
lan2_link_op_START_TIME43 := TARGET_TO_HOST(duration( 9.55000000000000E+00));
lan2_link_op_STOP_TIME43 := TARGET_TO_HOST(duration( 9.60000000000000E+00));
ctoc_op_START_TIME44 := TARGET_TO_HOST(duration( 9.60000000000000E+00));
ctoc_op_STOP_TIME44 := TARGET_TO_HOST(duration( 9.65000000000000E+00));
cnr_link_op_START_TIME45 := TARGET_TO_HOST(duration( 9.65000000000000E+00));
cnr_link_op_STOP_TIME45 := TARGET_TO_HOST(duration( 9.70000000000000E+00));
shooter_op_START_TIME46 := TARGET_TO_HOST(duration( 9.70000000000000E+00));
shooter_op_STOP_TIME46 := TARGET_TO_HOST(duration( 9.75000000000000E+00));
gui_input_event_monitor_START_TIME47 := TARGET_TO_HOST(duration( 9.75000000000000E+00));
gui_input_event_monitor_STOP_TIME47 := TARGET_TO_HOST(duration( 9.95000000000000E+00));
scdl_link_op_START_TIME48 := TARGET_TO_HOST(duration( 1.10000000000000E+01));
scdl_link_op_STOP_TIME48 := TARGET_TO_HOST(duration( 1.10500000000000E+01));
lan1_link_op_START_TIME49 := TARGET_TO_HOST(duration( 1.11000000000000E+01));
lan1_link_op_STOP_TIME49 := TARGET_TO_HOST(duration( 1.11500000000000E+01));
```

```
choose_inputs_START_TIME50 := TARGET_TO_HOST(duration( 1.13500000000000E+01));
choose_inputs_STOP_TIME50 := TARGET_TO_HOST(duration( 1.15500000000000E+01));
lan2_link_op_START_TIME51 := TARGET_TO_HOST(duration( 1.15500000000000E+01));
lan2_link_op_STOP_TIME51 := TARGET_TO_HOST(duration( 1.16000000000000E+01));
ctoc_op_START_TIME52 := TARGET_TO_HOST(duration( 1.16000000000000E+01));
ctoc_op_STOP_TIME52 := TARGET_TO_HOST(duration( 1.16500000000000E+01));
cnr_link_op_START_TIME53 := TARGET_TO_HOST(duration( 1.16500000000000E+01));
cnr_link_op_STOP_TIME53 := TARGET_TO_HOST(duration( 1.17000000000000E+01));
shooter_op_START_TIME54 := TARGET_TO_HOST(duration( 1.17000000000000E+01));
shooter_op_STOP_TIME54 := TARGET_TO_HOST(duration( 1.17500000000000E+01));
scdl_link_op_START_TIME55 := TARGET_TO_HOST(duration( 1.30000000000000E+01));
scdl_link_op_STOP_TIME55 := TARGET_TO_HOST(duration( 1.30500000000000E+01));
grnd_stat_mod_op_START_TIME56 := TARGET_TO_HOST(duration( 1.30500000000000E+01));
grnd_stat_mod_op_STOP_TIME56 := TARGET_TO_HOST(duration( 1.31000000000000E+01));
lan1_link_op_START_TIME57 := TARGET_TO_HOST(duration( 1.31000000000000E+01));
lan1_link_op_STOP_TIME57 := TARGET_TO_HOST(duration( 1.31500000000000E+01));
asas_op_START_TIME58 := TARGET_TO_HOST(duration( 1.31500000000000E+01));
asas_op_STOP_TIME58 := TARGET_TO_HOST(duration( 1.33500000000000E+01));
choose_inputs_START_TIME59 := TARGET_TO_HOST(duration( 1.33500000000000E+01));
choose_inputs_STOP_TIME59 := TARGET_TO_HOST(duration( 1.35500000000000E+01));
lan2_link_op_START_TIME60 := TARGET_TO_HOST(duration( 1.35500000000000E+01));
lan2_link_op_STOP_TIME60 := TARGET_TO_HOST(duration( 1.36000000000000E+01));
ctoc_op_START_TIME61 := TARGET_TO_HOST(duration( 1.36000000000000E+01));
ctoc_op_STOP_TIME61 := TARGET_TO_HOST(duration( 1.36500000000000E+01));
cnr_link_op_START_TIME62 := TARGET_TO_HOST(duration( 1.36500000000000E+01));
cnr_link_op_STOP_TIME62 := TARGET_TO_HOST(duration( 1.37000000000000E+01));
shooter_op_START_TIME63 := TARGET_TO_HOST(duration( 1.37000000000000E+01));
shooter_op_STOP_TIME63 := TARGET_TO_HOST(duration( 1.37500000000000E+01));
gui_input_event_monitor_START_TIME64 := TARGET_TO_HOST(duration( 1.37500000000000E+01));
gui_input_event_monitor_STOP_TIME64 := TARGET_TO_HOST(duration( 1.39500000000000E+01));
scdl_link_op_START_TIME65 := TARGET_TO_HOST(duration( 1.50000000000000E+01));
scdl_link_op_STOP_TIME65 := TARGET_TO_HOST(duration( 1.50500000000000E+01));
lan1_link_op_START_TIME66 := TARGET_TO_HOST(duration( 1.51000000000000E+01));
lan1_link_op_STOP_TIME66 := TARGET_TO_HOST(duration( 1.51500000000000E+01));
choose_inputs_START_TIME67 := TARGET_TO_HOST(duration( 1.53500000000000E+01));
choose_inputs_STOP_TIME67 := TARGET_TO_HOST(duration( 1.55500000000000E+01));
lan2_link_op_START_TIME68 := TARGET_TO_HOST(duration( 1.55500000000000E+01));
lan2_link_op_STOP_TIME68 := TARGET_TO_HOST(duration( 1.56000000000000E+01));
ctoc_op_START_TIME69 := TARGET_TO_HOST(duration( 1.56000000000000E+01));
ctoc_op_STOP_TIME69 := TARGET_TO_HOST(duration( 1.56500000000000E+01));
cnr_link_op_START_TIME70 := TARGET_TO_HOST(duration( 1.56500000000000E+01));
cnr_link_op_STOP_TIME70 := TARGET_TO_HOST(duration( 1.57000000000000E+01));
shooter_op_START_TIME71 := TARGET_TO_HOST(duration( 1.57000000000000E+01));
shooter_op_STOP_TIME71 := TARGET_TO_HOST(duration( 1.57500000000000E+01));
START(schedule_timer);
loop
 delay(target_emitter_op_START_TIME1 - HOST_DURATION(schedule_timer));
 target_emitter_op_DRIVER;
 if HOST_DURATION(schedule_timer) > target_emitter_op_STOP_TIME1 then
  PUT_LINE("timing error from operator target_emitter_op");
  SUBTRACT_HOST_TIME_FROM_ALL_TIMERS(HOST_DURATION(schedule_timer) - target_emitter_op_STOP_TIME1);
 end if;

 delay(jstars_op_START_TIME2 - HOST_DURATION(schedule_timer));
 jstars_op_DRIVER;
 if HOST_DURATION(schedule_timer) > jstars_op_STOP_TIME2 then
  PUT_LINE("timing error from operator jstars_op");
  SUBTRACT_HOST_TIME_FROM_ALL_TIMERS(HOST_DURATION(schedule_timer) - jstars_op_STOP_TIME2);
 end if;

 delay(scdl_link_op_START_TIME3 - HOST_DURATION(schedule_timer));
 scdl_link_op_DRIVER;
 if HOST_DURATION(schedule_timer) > scdl_link_op_STOP_TIME3 then
  PUT_LINE("timing error from operator scdl_link_op");
  SUBTRACT_HOST_TIME_FROM_ALL_TIMERS(HOST_DURATION(schedule_timer) - scdl_link_op_STOP_TIME3);
 end if;
```

```
delay(grnd_stat_mod_op_START_TIME4 - HOST_DURATION(schedule_timer));
grnd_stat_mod_op_DRIVER;
if HOST_DURATION(schedule_timer) > grnd_stat_mod_op_STOP_TIME4 then
  PUT_LINE("timing error from operator grnd_stat_mod_op");
  SUBTRACT_HOST_TIME_FROM_ALL_TIMERS(HOST_DURATION(schedule_timer) - grnd_stat_mod_op_STOP_TIME4);
end if;

delay(lan1_link_op_START_TIME5 - HOST_DURATION(schedule_timer));
lan1_link_op_DRIVER;
if HOST_DURATION(schedule_timer) > lan1_link_op_STOP_TIME5 then
  PUT_LINE("timing error from operator lan1_link_op");
  SUBTRACT_HOST_TIME_FROM_ALL_TIMERS(HOST_DURATION(schedule_timer) - lan1_link_op_STOP_TIME5);
end if;

delay(asas_op_START_TIME6 - HOST_DURATION(schedule_timer));
asas_op_DRIVER;
if HOST_DURATION(schedule_timer) > asas_op_STOP_TIME6 then
  PUT_LINE("timing error from operator asas_op");
  SUBTRACT_HOST_TIME_FROM_ALL_TIMERS(HOST_DURATION(schedule_timer) - asas_op_STOP_TIME6);
end if;

delay(choose_inputs_START_TIME7 - HOST_DURATION(schedule_timer));
choose_inputs_DRIVER;
if HOST_DURATION(schedule_timer) > choose_inputs_STOP_TIME7 then
  PUT_LINE("timing error from operator choose_inputs");
  SUBTRACT_HOST_TIME_FROM_ALL_TIMERS(HOST_DURATION(schedule_timer) - choose_inputs_STOP_TIME7);
end if;

delay(lan2_link_op_START_TIME8 - HOST_DURATION(schedule_timer));
lan2_link_op_DRIVER;
if HOST_DURATION(schedule_timer) > lan2_link_op_STOP_TIME8 then
  PUT_LINE("timing error from operator lan2_link_op");
  SUBTRACT_HOST_TIME_FROM_ALL_TIMERS(HOST_DURATION(schedule_timer) - lan2_link_op_STOP_TIME8);
end if;

delay(ctoc_op_START_TIME9 - HOST_DURATION(schedule_timer));
ctoc_op_DRIVER;
if HOST_DURATION(schedule_timer) > ctoc_op_STOP_TIME9 then
  PUT_LINE("timing error from operator ctoc_op");
  SUBTRACT_HOST_TIME_FROM_ALL_TIMERS(HOST_DURATION(schedule_timer) - ctoc_op_STOP_TIME9);
end if;

delay(cnr_link_op_START_TIME10 - HOST_DURATION(schedule_timer));
cnr_link_op_DRIVER;
if HOST_DURATION(schedule_timer) > cnr_link_op_STOP_TIME10 then
  PUT_LINE("timing error from operator cnr_link_op");
  SUBTRACT_HOST_TIME_FROM_ALL_TIMERS(HOST_DURATION(schedule_timer) - cnr_link_op_STOP_TIME10);
end if;

delay(shooter_op_START_TIME11 - HOST_DURATION(schedule_timer));
shooter_op_DRIVER;
if HOST_DURATION(schedule_timer) > shooter_op_STOP_TIME11 then
  PUT_LINE("timing error from operator shooter_op");
  SUBTRACT_HOST_TIME_FROM_ALL_TIMERS(HOST_DURATION(schedule_timer) - shooter_op_STOP_TIME11);
end if;

delay(gui_input_event_monitor_START_TIME12 - HOST_DURATION(schedule_timer));
gui_input_event_monitor_DRIVER;
if HOST_DURATION(schedule_timer) > gui_input_event_monitor_STOP_TIME12 then
  PUT_LINE("timing error from operator gui_input_event_monitor");
  SUBTRACT_HOST_TIME_FROM_ALL_TIMERS(HOST_DURATION(schedule_timer) -
  gui_input_event_monitor_STOP_TIME12);
end if;
```

```
delay(scdl_link_op_START_TIME13 - HOST_DURATION(schedule_timer));
scdl_link_op_DRIVER;
if HOST_DURATION(schedule_timer) > scdl_link_op_STOP_TIME13 then
 PUT_LINE("timing error from operator scdl_link_op");
 SUBTRACT_HOST_TIME_FROM_ALL_TIMERS(HOST_DURATION(schedule_timer) - scdl_link_op_STOP_TIME13);
end if;

delay(lan1_link_op_START_TIME14 - HOST_DURATION(schedule_timer));
lan1_link_op_DRIVER;
if HOST_DURATION(schedule_timer) > lan1_link_op_STOP_TIME14 then
 PUT_LINE("timing error from operator lan1_link_op");
 SUBTRACT_HOST_TIME_FROM_ALL_TIMERS(HOST_DURATION(schedule_timer) - lan1_link_op_STOP_TIME14);
end if;

delay(choose_inputs_START_TIME15 - HOST_DURATION(schedule_timer));
choose_inputs_DRIVER;
if HOST_DURATION(schedule_timer) > choose_inputs_STOP_TIME15 then
 PUT_LINE("timing error from operator choose_inputs");
 SUBTRACT_HOST_TIME_FROM_ALL_TIMERS(HOST_DURATION(schedule_timer) - choose_inputs_STOP_TIME15);
end if;

delay(lan2_link_op_START_TIME16 - HOST_DURATION(schedule_timer));
lan2_link_op_DRIVER;
if HOST_DURATION(schedule_timer) > lan2_link_op_STOP_TIME16 then
 PUT_LINE("timing error from operator lan2_link_op");
 SUBTRACT_HOST_TIME_FROM_ALL_TIMERS(HOST_DURATION(schedule_timer) - lan2_link_op_STOP_TIME16);
end if;

delay(ctoc_op_START_TIME17 - HOST_DURATION(schedule_timer));
ctoc_op_DRIVER;
if HOST_DURATION(schedule_timer) > ctoc_op_STOP_TIME17 then
 PUT_LINE("timing error from operator ctoc_op");
 SUBTRACT_HOST_TIME_FROM_ALL_TIMERS(HOST_DURATION(schedule_timer) - ctoc_op_STOP_TIME17);
end if;

delay(cnr_link_op_START_TIME18 - HOST_DURATION(schedule_timer));
cnr_link_op_DRIVER;
if HOST_DURATION(schedule_timer) > cnr_link_op_STOP_TIME18 then
 PUT_LINE("timing error from operator cnr_link_op");
 SUBTRACT_HOST_TIME_FROM_ALL_TIMERS(HOST_DURATION(schedule_timer) - cnr_link_op_STOP_TIME18);
end if;

delay(shooter_op_START_TIME19 - HOST_DURATION(schedule_timer));
shooter_op_DRIVER;
if HOST_DURATION(schedule_timer) > shooter_op_STOP_TIME19 then
 PUT_LINE("timing error from operator shooter_op");
 SUBTRACT_HOST_TIME_FROM_ALL_TIMERS(HOST_DURATION(schedule_timer) - shooter_op_STOP_TIME19);
end if;

delay(scdl_link_op_START_TIME20 - HOST_DURATION(schedule_timer));
scdl_link_op_DRIVER;
if HOST_DURATION(schedule_timer) > scdl_link_op_STOP_TIME20 then
 PUT_LINE("timing error from operator scdl_link_op");
 SUBTRACT_HOST_TIME_FROM_ALL_TIMERS(HOST_DURATION(schedule_timer) - scdl_link_op_STOP_TIME20);
end if;

delay(grnd_stat_mod_op_START_TIME21 - HOST_DURATION(schedule_timer));
grnd_stat_mod_op_DRIVER;
if HOST_DURATION(schedule_timer) > grnd_stat_mod_op_STOP_TIME21 then
 PUT_LINE("timing error from operator grnd_stat_mod_op");
 SUBTRACT_HOST_TIME_FROM_ALL_TIMERS(HOST_DURATION(schedule_timer) -grnd_stat_mod_op_STOP_TIME21);
end if;

delay(lan1_link_op_START_TIME22 - HOST_DURATION(schedule_timer));
lan1_link_op_DRIVER;
if HOST_DURATION(schedule_timer) > lan1_link_op_STOP_TIME22 then
 PUT_LINE("timing error from operator lan1_link_op");
 SUBTRACT_HOST_TIME_FROM_ALL_TIMERS(HOST_DURATION(schedule_timer) - lan1_link_op_STOP_TIME22);
end if;
```

```
delay(asas_op_START_TIME23 - HOST_DURATION(schedule_timer));
asas_op_DRIVER;
if HOST_DURATION(schedule_timer) > asas_op_STOP_TIME23 then
  PUT_LINE("timing error from operator asas_op");
  SUBTRACT_HOST_TIME_FROM_ALL_TIMERS(HOST_DURATION(schedule_timer) - asas_op_STOP_TIME23);
end if;

delay(choose_inputs_START_TIME24 - HOST_DURATION(schedule_timer));
choose_inputs_DRIVER;
if HOST_DURATION(schedule_timer) > choose_inputs_STOP_TIME24 then
  PUT_LINE("timing error from operator choose_inputs");
  SUBTRACT_HOST_TIME_FROM_ALL_TIMERS(HOST_DURATION(schedule_timer) - choose_inputs_STOP_TIME24);
end if;

delay(lan2_link_op_START_TIME25 - HOST_DURATION(schedule_timer));
lan2_link_op_DRIVER;
if HOST_DURATION(schedule_timer) > lan2_link_op_STOP_TIME25 then
  PUT_LINE("timing error from operator lan2_link_op");
  SUBTRACT_HOST_TIME_FROM_ALL_TIMERS(HOST_DURATION(schedule_timer) - lan2_link_op_STOP_TIME25);
end if;

delay(ctoc_op_START_TIME26 - HOST_DURATION(schedule_timer));
ctoc_op_DRIVER;
if HOST_DURATION(schedule_timer) > ctoc_op_STOP_TIME26 then
  PUT_LINE("timing error from operator ctoc_op");
  SUBTRACT_HOST_TIME_FROM_ALL_TIMERS(HOST_DURATION(schedule_timer) - ctoc_op_STOP_TIME26);
end if;

delay(cnr_link_op_START_TIME27 - HOST_DURATION(schedule_timer));
cnr_link_op_DRIVER;
if HOST_DURATION(schedule_timer) > cnr_link_op_STOP_TIME27 then
  PUT_LINE("timing error from operator cnr_link_op");
  SUBTRACT_HOST_TIME_FROM_ALL_TIMERS(HOST_DURATION(schedule_timer) - cnr_link_op_STOP_TIME27);
end if;

delay(shooter_op_START_TIME28 - HOST_DURATION(schedule_timer));
shooter_op_DRIVER;
if HOST_DURATION(schedule_timer) > shooter_op_STOP_TIME28 then
  PUT_LINE("timing error from operator shooter_op");
  SUBTRACT_HOST_TIME_FROM_ALL_TIMERS(HOST_DURATION(schedule_timer) - shooter_op_STOP_TIME28);
end if;

delay(gui_input_event_monitor_START_TIME29 - HOST_DURATION(schedule_timer));
gui_input_event_monitor_DRIVER;
if HOST_DURATION(schedule_timer) > gui_input_event_monitor_STOP_TIME29 then
  PUT_LINE("timing error from operator gui_input_event_monitor");
  SUBTRACT_HOST_TIME_FROM_ALL_TIMERS(HOST_DURATION(schedule_timer) -
  gui_input_event_monitor_STOP_TIME29);
end if;

delay(scdl_link_op_START_TIME30 - HOST_DURATION(schedule_timer));
scdl_link_op_DRIVER;
if HOST_DURATION(schedule_timer) > scdl_link_op_STOP_TIME30 then
  PUT_LINE("timing error from operator scdl_link_op");
  SUBTRACT_HOST_TIME_FROM_ALL_TIMERS(HOST_DURATION(schedule_timer) - scdl_link_op_STOP_TIME30);
end if;

delay(lan1_link_op_START_TIME31 - HOST_DURATION(schedule_timer));
lan1_link_op_DRIVER;
if HOST_DURATION(schedule_timer) > lan1_link_op_STOP_TIME31 then
  PUT_LINE("timing error from operator lan1_link_op");
  SUBTRACT_HOST_TIME_FROM_ALL_TIMERS(HOST_DURATION(schedule_timer) - lan1_link_op_STOP_TIME31);
end if;

delay(choose_inputs_START_TIME32 - HOST_DURATION(schedule_timer));
choose_inputs_DRIVER;
if HOST_DURATION(schedule_timer) > choose_inputs_STOP_TIME32 then
  PUT_LINE("timing error from operator choose_inputs");
```

```
 SUBTRACT_HOST_TIME_FROM_ALL_TIMERS(HOST_DURATION(schedule_timer) - choose_inputs_STOP_TIME32);
end if;

delay(lan2_link_op_START_TIME33 - HOST_DURATION(schedule_timer));
lan2_link_op_DRIVER;
if HOST_DURATION(schedule_timer) > lan2_link_op_STOP_TIME33 then
 PUT_LINE("timing error from operator lan2_link_op");
 SUBTRACT_HOST_TIME_FROM_ALL_TIMERS(HOST_DURATION(schedule_timer) - lan2_link_op_STOP_TIME33);
end if;

delay(ctoc_op_START_TIME34 - HOST_DURATION(schedule_timer));
ctoc_op_DRIVER;
if HOST_DURATION(schedule_timer) > ctoc_op_STOP_TIME34 then
 PUT_LINE("timing error from operator ctoc_op");
 SUBTRACT_HOST_TIME_FROM_ALL_TIMERS(HOST_DURATION(schedule_timer) - ctoc_op_STOP_TIME34);
end if;

delay(cnr_link_op_START_TIME35 - HOST_DURATION(schedule_timer));
cnr_link_op_DRIVER;
if HOST_DURATION(schedule_timer) > cnr_link_op_STOP_TIME35 then
 PUT_LINE("timing error from operator cnr_link_op");
 SUBTRACT_HOST_TIME_FROM_ALL_TIMERS(HOST_DURATION(schedule_timer) - cnr_link_op_STOP_TIME35);
end if;

delay(shooter_op_START_TIME36 - HOST_DURATION(schedule_timer));
shooter_op_DRIVER;
if HOST_DURATION(schedule_timer) > shooter_op_STOP_TIME36 then
 PUT_LINE("timing error from operator shooter_op");
 SUBTRACT_HOST_TIME_FROM_ALL_TIMERS(HOST_DURATION(schedule_timer) - shooter_op_STOP_TIME36);
end if;

delay(jstars_op_START_TIME37 - HOST_DURATION(schedule_timer));
jstars_op_DRIVER;
if HOST_DURATION(schedule_timer) > jstars_op_STOP_TIME37 then
 PUT_LINE("timing error from operator jstars_op");
 SUBTRACT_HOST_TIME_FROM_ALL_TIMERS(HOST_DURATION(schedule_timer) - jstars_op_STOP_TIME37);
end if;

delay(scdl_link_op_START_TIME38 - HOST_DURATION(schedule_timer));
scdl_link_op_DRIVER;
if HOST_DURATION(schedule_timer) > scdl_link_op_STOP_TIME38 then
 PUT_LINE("timing error from operator scdl_link_op");
 SUBTRACT_HOST_TIME_FROM_ALL_TIMERS(HOST_DURATION(schedule_timer) - scdl_link_op_STOP_TIME38);
end if;

delay(grnd_stat_mod_op_START_TIME39 - HOST_DURATION(schedule_timer));
grnd_stat_mod_op_DRIVER;
if HOST_DURATION(schedule_timer) > grnd_stat_mod_op_STOP_TIME39 then
 PUT_LINE("timing error from operator grnd_stat_mod_op");
 SUBTRACT_HOST_TIME_FROM_ALL_TIMERS(HOST_DURATION(schedule_timer) -
 grnd_stat_mod_op_STOP_TIME39);
end if;

delay(lan1_link_op_START_TIME40 - HOST_DURATION(schedule_timer));
lan1_link_op_DRIVER;
if HOST_DURATION(schedule_timer) > lan1_link_op_STOP_TIME40 then
 PUT_LINE("timing error from operator lan1_link_op");
 SUBTRACT_HOST_TIME_FROM_ALL_TIMERS(HOST_DURATION(schedule_timer) - lan1_link_op_STOP_TIME40);
end if;

delay(asas_op_START_TIME41 - HOST_DURATION(schedule_timer));
asas_op_DRIVER;
if HOST_DURATION(schedule_timer) > asas_op_STOP_TIME41 then
 PUT_LINE("timing error from operator asas_op");
 SUBTRACT_HOST_TIME_FROM_ALL_TIMERS(HOST_DURATION(schedule_timer) - asas_op_STOP_TIME41);
end if;
```

```
delay(choose_inputs_START_TIME42 - HOST_DURATION(schedule_timer));
choose_inputs_DRIVER;
if HOST_DURATION(schedule_timer) > choose_inputs_STOP_TIME42 then
 PUT_LINE("timing error from operator choose_inputs");
 SUBTRACT_HOST_TIME_FROM_ALL_TIMERS(HOST_DURATION(schedule_timer) - choose_inputs_STOP_TIME42);
end if;

delay(lan2_link_op_START_TIME43 - HOST_DURATION(schedule_timer));
lan2_link_op_DRIVER;
if HOST_DURATION(schedule_timer) > lan2_link_op_STOP_TIME43 then
 PUT_LINE("timing error from operator lan2_link_op");
 SUBTRACT_HOST_TIME_FROM_ALL_TIMERS(HOST_DURATION(schedule_timer) - lan2_link_op_STOP_TIME43);
end if;

delay(ctoc_op_START_TIME44 - HOST_DURATION(schedule_timer));
ctoc_op_DRIVER;
if HOST_DURATION(schedule_timer) > ctoc_op_STOP_TIME44 then
 PUT_LINE("timing error from operator ctoc_op");
 SUBTRACT_HOST_TIME_FROM_ALL_TIMERS(HOST_DURATION(schedule_timer) - ctoc_op_STOP_TIME44);
end if;

delay(cnr_link_op_START_TIME45 - HOST_DURATION(schedule_timer));
cnr_link_op_DRIVER;
if HOST_DURATION(schedule_timer) > cnr_link_op_STOP_TIME45 then
 PUT_LINE("timing error from operator cnr_link_op");
 SUBTRACT_HOST_TIME_FROM_ALL_TIMERS(HOST_DURATION(schedule_timer) - cnr_link_op_STOP_TIME45);
end if;

delay(shooter_op_START_TIME46 - HOST_DURATION(schedule_timer));
shooter_op_DRIVER;
if HOST_DURATION(schedule_timer) > shooter_op_STOP_TIME46 then
 PUT_LINE("timing error from operator shooter_op");
 SUBTRACT_HOST_TIME_FROM_ALL_TIMERS(HOST_DURATION(schedule_timer) - shooter_op_STOP_TIME46);
end if;

delay(gui_input_event_monitor_START_TIME47 - HOST_DURATION(schedule_timer));
gui_input_event_monitor_DRIVER;
if HOST_DURATION(schedule_timer) > gui_input_event_monitor_STOP_TIME47 then
 PUT_LINE("timing error from operator gui_input_event_monitor");
 SUBTRACT_HOST_TIME_FROM_ALL_TIMERS(HOST_DURATION(schedule_timer) -
 gui_input_event_monitor_STOP_TIME47);
end if;

delay(scdl_link_op_START_TIME48 - HOST_DURATION(schedule_timer));
scdl_link_op_DRIVER;
if HOST_DURATION(schedule_timer) > scdl_link_op_STOP_TIME48 then
 PUT_LINE("timing error from operator scdl_link_op");
 SUBTRACT_HOST_TIME_FROM_ALL_TIMERS(HOST_DURATION(schedule_timer) - scdl_link_op_STOP_TIME48);
end if;

delay(lan1_link_op_START_TIME49 - HOST_DURATION(schedule_timer));
lan1_link_op_DRIVER;
if HOST_DURATION(schedule_timer) > lan1_link_op_STOP_TIME49 then
 PUT_LINE("timing error from operator lan1_link_op");
 SUBTRACT_HOST_TIME_FROM_ALL_TIMERS(HOST_DURATION(schedule_timer) - lan1_link_op_STOP_TIME49);
end if;

delay(choose_inputs_START_TIME50 - HOST_DURATION(schedule_timer));
choose_inputs_DRIVER;
if HOST_DURATION(schedule_timer) > choose_inputs_STOP_TIME50 then
 PUT_LINE("timing error from operator choose_inputs");
 SUBTRACT_HOST_TIME_FROM_ALL_TIMERS(HOST_DURATION(schedule_timer) - choose_inputs_STOP_TIME50);
end if;

delay(lan2_link_op_START_TIME51 - HOST_DURATION(schedule_timer));
lan2_link_op_DRIVER;
if HOST_DURATION(schedule_timer) > lan2_link_op_STOP_TIME51 then
 PUT_LINE("timing error from operator lan2_link_op");
 SUBTRACT_HOST_TIME_FROM_ALL_TIMERS(HOST_DURATION(schedule_timer) - lan2_link_op_STOP_TIME51);
```

```
end if;

delay(ctoc_op_START_TIME52 - HOST_DURATION(schedule_timer));
ctoc_op_DRIVER;
if HOST_DURATION(schedule_timer) > ctoc_op_STOP_TIME52 then
 PUT_LINE("timing error from operator ctoc_op");
 SUBTRACT_HOST_TIME_FROM_ALL_TIMERS(HOST_DURATION(schedule_timer) - ctoc_op_STOP_TIME52);
end if;

delay(cnr_link_op_START_TIME53 - HOST_DURATION(schedule_timer));
cnr_link_op_DRIVER;
if HOST_DURATION(schedule_timer) > cnr_link_op_STOP_TIME53 then
 PUT_LINE("timing error from operator cnr_link_op");
 SUBTRACT_HOST_TIME_FROM_ALL_TIMERS(HOST_DURATION(schedule_timer) - cnr_link_op_STOP_TIME53);
end if;

delay(shooter_op_START_TIME54 - HOST_DURATION(schedule_timer));
shooter_op_DRIVER;
if HOST_DURATION(schedule_timer) > shooter_op_STOP_TIME54 then
 PUT_LINE("timing error from operator shooter_op");
 SUBTRACT_HOST_TIME_FROM_ALL_TIMERS(HOST_DURATION(schedule_timer) - shooter_op_STOP_TIME54);
end if;

delay(scdl_link_op_START_TIME55 - HOST_DURATION(schedule_timer));
scdl_link_op_DRIVER;
if HOST_DURATION(schedule_timer) > scdl_link_op_STOP_TIME55 then
 PUT_LINE("timing error from operator scdl_link_op");
 SUBTRACT_HOST_TIME_FROM_ALL_TIMERS(HOST_DURATION(schedule_timer) - scdl_link_op_STOP_TIME55);
end if;

delay(grnd_stat_mod_op_START_TIME56 - HOST_DURATION(schedule_timer));
grnd_stat_mod_op_DRIVER;
if HOST_DURATION(schedule_timer) > grnd_stat_mod_op_STOP_TIME56 then
 PUT_LINE("timing error from operator grnd_stat_mod_op");
 SUBTRACT_HOST_TIME_FROM_ALL_TIMERS(HOST_DURATION(schedule_timer) -
 grnd_stat_mod_op_STOP_TIME56);
end if;

delay(lan1_link_op_START_TIME57 - HOST_DURATION(schedule_timer));
lan1_link_op_DRIVER;
if HOST_DURATION(schedule_timer) > lan1_link_op_STOP_TIME57 then
 PUT_LINE("timing error from operator lan1_link_op");
 SUBTRACT_HOST_TIME_FROM_ALL_TIMERS(HOST_DURATION(schedule_timer) - lan1_link_op_STOP_TIME57);
end if;

delay(asas_op_START_TIME58 - HOST_DURATION(schedule_timer));
asas_op_DRIVER;
if HOST_DURATION(schedule_timer) > asas_op_STOP_TIME58 then
 PUT_LINE("timing error from operator asas_op");
 SUBTRACT_HOST_TIME_FROM_ALL_TIMERS(HOST_DURATION(schedule_timer) - asas_op_STOP_TIME58);
end if;

delay(choose_inputs_START_TIME59 - HOST_DURATION(schedule_timer));
choose_inputs_DRIVER;
if HOST_DURATION(schedule_timer) > choose_inputs_STOP_TIME59 then
 PUT_LINE("timing error from operator choose_inputs");
 SUBTRACT_HOST_TIME_FROM_ALL_TIMERS(HOST_DURATION(schedule_timer) - choose_inputs_STOP_TIME59);
end if;

delay(lan2_link_op_START_TIME60 - HOST_DURATION(schedule_timer));
lan2_link_op_DRIVER;
if HOST_DURATION(schedule_timer) > lan2_link_op_STOP_TIME60 then
 PUT_LINE("timing error from operator lan2_link_op");
 SUBTRACT_HOST_TIME_FROM_ALL_TIMERS(HOST_DURATION(schedule_timer) - lan2_link_op_STOP_TIME60);
end if;

delay(ctoc_op_START_TIME61 - HOST_DURATION(schedule_timer));
ctoc_op_DRIVER;
if HOST_DURATION(schedule_timer) > ctoc_op_STOP_TIME61 then
```

```
  PUT_LINE("timing error from operator ctoc_op");
  SUBTRACT_HOST_TIME_FROM_ALL_TIMERS(HOST_DURATION(schedule_timer) - ctoc_op_STOP_TIME61);
end if;

delay(cnr_link_op_START_TIME62 - HOST_DURATION(schedule_timer));
cnr_link_op_DRIVER;
if HOST_DURATION(schedule_timer) > cnr_link_op_STOP_TIME62 then
  PUT_LINE("timing error from operator cnr_link_op");
  SUBTRACT_HOST_TIME_FROM_ALL_TIMERS(HOST_DURATION(schedule_timer) - cnr_link_op_STOP_TIME62);
end if;

delay(shooter_op_START_TIME63 - HOST_DURATION(schedule_timer));
shooter_op_DRIVER;
if HOST_DURATION(schedule_timer) > shooter_op_STOP_TIME63 then
  PUT_LINE("timing error from operator shooter_op");
  SUBTRACT_HOST_TIME_FROM_ALL_TIMERS(HOST_DURATION(schedule_timer) - shooter_op_STOP_TIME63);
end if;

delay(gui_input_event_monitor_START_TIME64 - HOST_DURATION(schedule_timer));
gui_input_event_monitor_DRIVER;
if HOST_DURATION(schedule_timer) > gui_input_event_monitor_STOP_TIME64 then
  PUT_LINE("timing error from operator gui_input_event_monitor");
  SUBTRACT_HOST_TIME_FROM_ALL_TIMERS(HOST_DURATION(schedule_timer) -
  gui_input_event_monitor_STOP_TIME64);
end if;

delay(scdl_link_op_START_TIME65 - HOST_DURATION(schedule_timer));
scdl_link_op_DRIVER;
if HOST_DURATION(schedule_timer) > scdl_link_op_STOP_TIME65 then
  PUT_LINE("timing error from operator scdl_link_op");
  SUBTRACT_HOST_TIME_FROM_ALL_TIMERS(HOST_DURATION(schedule_timer) - scdl_link_op_STOP_TIME65);
end if;

delay(lan1_link_op_START_TIME66 - HOST_DURATION(schedule_timer));
lan1_link_op_DRIVER;
if HOST_DURATION(schedule_timer) > lan1_link_op_STOP_TIME66 then
  PUT_LINE("timing error from operator lan1_link_op");
  SUBTRACT_HOST_TIME_FROM_ALL_TIMERS(HOST_DURATION(schedule_timer) - lan1_link_op_STOP_TIME66);
end if;

delay(choose_inputs_START_TIME67 - HOST_DURATION(schedule_timer));
choose_inputs_DRIVER;
if HOST_DURATION(schedule_timer) > choose_inputs_STOP_TIME67 then
  PUT_LINE("timing error from operator choose_inputs");
  SUBTRACT_HOST_TIME_FROM_ALL_TIMERS(HOST_DURATION(schedule_timer) - choose_inputs_STOP_TIME67);
end if;

delay(lan2_link_op_START_TIME68 - HOST_DURATION(schedule_timer));
lan2_link_op_DRIVER;
if HOST_DURATION(schedule_timer) > lan2_link_op_STOP_TIME68 then
  PUT_LINE("timing error from operator lan2_link_op");
  SUBTRACT_HOST_TIME_FROM_ALL_TIMERS(HOST_DURATION(schedule_timer) - lan2_link_op_STOP_TIME68);
end if;

delay(ctoc_op_START_TIME69 - HOST_DURATION(schedule_timer));
ctoc_op_DRIVER;
if HOST_DURATION(schedule_timer) > ctoc_op_STOP_TIME69 then
  PUT_LINE("timing error from operator ctoc_op");
  SUBTRACT_HOST_TIME_FROM_ALL_TIMERS(HOST_DURATION(schedule_timer) - ctoc_op_STOP_TIME69);
end if;

delay(cnr_link_op_START_TIME70 - HOST_DURATION(schedule_timer));
cnr_link_op_DRIVER;
if HOST_DURATION(schedule_timer) > cnr_link_op_STOP_TIME70 then
  PUT_LINE("timing error from operator cnr_link_op");
  SUBTRACT_HOST_TIME_FROM_ALL_TIMERS(HOST_DURATION(schedule_timer) - cnr_link_op_STOP_TIME70);
end if;

delay(shooter_op_START_TIME71 - HOST_DURATION(schedule_timer));
```

```
    shooter_op_DRIVER;
    if HOST_DURATION(schedule_timer) > shooter_op_STOP_TIME71 then
     PUT_LINE("timing error from operator shooter_op");
     SUBTRACT_HOST_TIME_FROM_ALL_TIMERS(HOST_DURATION(schedule_timer) - shooter_op_STOP_TIME71);
    end if;

    delay(PERIOD - HOST_DURATION(schedule_timer));
    RESET(schedule_timer);
   end loop;
  end STATIC_SCHEDULE_TYPE;

  procedure START_STATIC_SCHEDULE is
  begin
   STATIC_SCHEDULE.START;
  end START_STATIC_SCHEDULE;

end atacms_STATIC_SCHEDULERS;


 with ATACMS_STATIC_SCHEDULERS; use ATACMS_STATIC_SCHEDULERS;
 with ATACMS_DYNAMIC_SCHEDULERS; use ATACMS_DYNAMIC_SCHEDULERS;
 with CAPS_HARDWARE_MODEL; use CAPS_HARDWARE_MODEL;
procedure ATACMS is
begin
 init_hardware_model;
 start_static_schedule;
 start_dynamic_schedule;
end ATACMS;
```

atacms.asas_op.a

```
-- filename: atacms.asas_op.a
-- Created 05 Jun 96, mod 7/29/96;mod 04 Sep 96
-- Purpose:    This package simulates the operations of the ASAS.
-- This module takes the simulate video [array of records], looks for
--  targets, prioritizes them, then orders the shooter to fire at them one
--  at a time.
-- compile... ada atacms.asas_op.a

with text_io; use text_io;
with target_data_PKG;
with grnd_stat_mod_array_PKG;
with constants_PKG;
with asas_lv_array_PKG;
with my_unit_PKG;

package asas_op_PKG is

 procedure asas_op (target_array4_str : grnd_stat_mod_array_PKG.grnd_stat_mod_array;
             gui_in_str : my_unit_PKG.my_unit;
             fire_cmd1_str : out target_data_PKG.target_data);

end asas_op_PKG;



package body asas_op_PKG is

 -- This array tracks the fired on 'status' of the targets.  This precludes firing a second
 -- time at the same target
 --   a '0' indicates no target present and thus has not been fired on (i.e., null status
 --   a '1' indicates this target is awaiting an available shooter
 --   a '2' indicates a fire mission has been assigned to this target  - future use
 --   a '3' if implemented would indicate the target has been shot at & neutralized - future use
 -- The array index will be the 'tgt_num' of the IN parameter
 LV_been_shot_yet : asas_lv_array_PKG.asas_lv_array;

 -- used to track which targets have been assigned to the the shooter
 LV_counter : natural := 1;

 -- used to hold the last target number (index) examined during the previous running
 --  of this module
 LV_holder  : natural := 0;

 procedure asas_operator_delay is
 begin
   delay duration(constants_PKG.mean_asas_operator_delay); --temporary;use mean & deviation
                                                              --to calculate delay
                                                              --(see constants_PKG)
 end asas_operator_delay;

 procedure asas_processing_delay is
 begin
   delay duration(constants_PKG.mean_asas_processing_delay);--temporary;use mean & deviation
                                                              --to calculate delay(see
                                                              --constants_PKG)
 end asas_processing_delay;

 procedure asas_transmission_prep_delay is
 begin
   delay duration(constants_PKG.mean_asas_transmission_prep_delay);--temporary;use mean &
                                                                     --deviation to calculate
                                                                     --delay(see constants_PKG)
 end asas_transmission_prep_delay;
```

```
procedure asas_error is
begin
  null;                                  --insert simulated errors
end asas_error;

procedure asas_op (target_array4_str : grnd_stat_mod_array_PKG.grnd_stat_mod_array;
            gui_in_str :  my_unit_PKG.my_unit;
            fire_cmd1_str : out target_data_PKG.target_data) is

 index : natural := 0;  -- an local index to match the 'tgt_num' of the IN parameter
 row : natural := 0;
 col : natural := 0;

begin

 Put_line("ASAS processing targeting information...");

 -- loop thru IN parameter array of records looking for targets.
 -- Those records with a 'tgt_num' are targets
 for row_x in  constants_PKG.min_array..constants_PKG.max_array loop
   for column_y in constants_PKG.min_array..constants_PKG.max_array loop


         -- only do this (grant it an 'awaiting shooter' status) if it does have
         -- a previously assigned tgt_num AND it hasn't previously been assigned a status
         -- on a previous trip thru this 'if' statement
   if (target_array4_str(row_x,column_y).tgt_num /= 0) AND
            (LV_been_shot_yet(target_array4_str(row_x,column_y).tgt_num).status = 0) then

                  -- copy it here for readability below,
                  index := target_array4_str(row_x,column_y).tgt_num;

         -- Once you find a target, put it in an 'awaiting shooter' status
                  --   and record it's position for later use
                  LV_been_shot_yet(index).target_class :=
                              target_array4_str(row_x,column_y).target_class;
         LV_been_shot_yet(index).status := 1;
         LV_been_shot_yet(index).x_val := row_x;
                  LV_been_shot_yet(index).y_val := column_y;
                  LV_been_shot_yet(index).tgt_num := index;

                  -- for debugging*******
                  --put("In ASAS 'if', the current tgt_num is ");
       -- constants_PKG.int_io.put(index,0);
                  --put("  and LV_max_tgt_num..."); constants_PKG.int_io.put(LV_max_tgt_num,0);
                  --new_line;

   end if;
  end loop;
 end loop;

--*****************

 -- Next job is to send the next target to the shooter
 -- First, simulate prioritizing the current targets
 -- procedure prioritize(LV_been_shot_yet : in out asas_lv_array_PKG.asas_lv_array);

 -- Now, only enter this section if the counter below points to a record with a valid target
 if LV_been_shot_yet(LV_counter).status /= 0 then

         row := LV_been_shot_yet(LV_counter).x_val;
         col := LV_been_shot_yet(LV_counter).y_val;

   -- pass target to OUT parameter
   fire_cmd1_str.target_class :=
                  target_array4_str(row,col).target_class;
    fire_cmd1_str.easting :=
                  target_array4_str(row,col).easting;
   fire_cmd1_str.northing :=
                  target_array4_str(row,col).northing;
```

```
fire_cmd1_str.alt :=
                          target_array4_str(row,col).alt;
     fire_cmd1_str.tgt_num :=
                          target_array4_str(row,col).tgt_num;
     fire_cmd1_str.good_xmission :=
                   target_array4_str(constants_PKG.min_array,constants_PKG.min_array).good_xmission;
            fire_cmd1_str.status :=
                                  LV_been_shot_yet(LV_counter).status;
            LV_counter := LV_counter +1;   -- point to next target in the array

   end if;


--THESE ARE PLACEHOLDERS; USE/MODIFY AS NECESSARY
   --asas_operator_delay;
   --asas_processing_delay;
   --asas_transmission_prep_delay;
   --asas_error;



 end asas_op;

end asas_op_PKG;



--put("Good xmission so far? ... ");
--constants_PKG.bool_io.put(target_array4_str(constants_PKG.min_array,
--                       constants_PKG.min_array).good_xmission); new_line;


--put("In ASAS, LV_counter... ");
--constants_PKG.int_io.put(LV_counter,0);  new_line;  -- for debugging
```

atacms.choose_inputs.a

```
-- filename: mult_weather.choose_inputs.a

--
-- Authors:  Maj George Whitbeck and LCDR David Angrisani
-- Date:     6 Aug 96
-- Project:  Thesis - A CAPS Prototype of the ATACMS C3 System
-- Purpose:

-- a child bubble of parent "gui_in" from our thesis
-- This file is a candidate for auto code generation in the future.
-- Dr Berzin's script doesn't currently generate this but it could

with my_unit_pkg;  -- provides a type of "Start, Pause, Add_Target, Quit"
with text_io; use text_io;

package choose_inputs_PKG is

     procedure choose_inputs(gui_in_str : out my_unit_PKG.my_unit);

     -- procedure "record_input" is used as follows.
     -- User selects "Start, Pause, Add_Target, or Quit" with mouse.
     -- Event handler in package "atacms.pan_gui_in_b.a" reads the mouse
     -- event and makes a procedure call to this procedure.  It sends along either
     -- the words Start, Pause, Add_Target, or Quit".
     procedure record_input(gui_in_str : in my_unit_PKG.my_unit);

end choose_inputs_PKG;

with psdl_streams; use psdl_streams;

package body choose_inputs_PKG is

     --******
     package gui_in_str_buffer is new sampled_buffer(my_unit_PKG.my_unit);
     use gui_in_str_buffer;

     --*******
     -- can cut and paste the guts of this for any app, change parameter, etc
     procedure choose_inputs(gui_in_str : out my_unit_PKG.my_unit) is
     begin
          -- below is from gui_in_str_buffer
          buffer.read(gui_in_str);
          --put_line("buffer read, in choose_inputs");

     end choose_inputs;

     --*******
     -- can cut and paste the guts of this for any app, change parameter, etc
     procedure record_input(gui_in_str : in my_unit_PKG.my_unit) IS
     begin
          -- below is from gui_in_str_buffer
          buffer.write(gui_in_str);
          --put_line("Buffer write-- in choose_inputs.a, radio button pushed.");

     end record_input;

end choose_inputs_PKG;
```

atacms.cmds_out.a

```
-- filename: atacms.cmds_out.a
--
-- Authors:  Maj George Whitbeck and LCDR David Angrisani
-- Date:     6 Aug 96
-- Project:  Thesis - A CAPS Prototype of the ATACMS C3 System
-- Purpose:  This is the matching ada file for the output items specified
--     as a TAE item in panel "gui_out", and as a child bubble of
--     parent "gui_out"


with text_io; use text_io;
with target_data_PKG; --### manually added


package cmds_out_pkg is
  procedure cmds_out(gui_out_str: target_data_PKG.target_data);
end cmds_out_pkg;

  with atacms_input_event_monitor_task_pkg;
  use atacms_input_event_monitor_task_pkg;
package body cmds_out_pkg is
  procedure cmds_out(gui_out_str: target_data_PKG.target_data) is
  begin
--put_line("^^^^about to request a rendezvous... in cmds_out.a  ");

    atacms_input_event_monitor_task.cmds_out_entry(gui_out_str);
--put_line("^^^^^just retruned from a rendezvous... in cmds_out.a  ");

  end cmds_out;
end cmds_out_pkg;
```

atacms.cnr_link_op.a

```
-- filename: atacms.cnr_link_op.a
--
-- Authors:  Maj George Whitbeck and LCDR David Angrisani
-- Date:     6 Aug 96;mod 04 Sep 96
-- Project:  Thesis - A CAPS Prototype of the ATACMS C3 System
-- Purpose:

-- this comm link will pass on the data 90% of the time.
-- In 10% of the transmissions,
--   we simulate friction and the mission dies in the network

-- to compile... ada atacms.cnr_link_op.a


with text_io; use text_io;
with target_data_PKG;
with global_random_PKG;
with constants_PKG;

package cnr_link_op_PKG is
 procedure cnr_link_op (fire_cmd3_str : in target_data_PKG.target_data;
               fire_cmd4_str : out target_data_PKG.target_data );
end cnr_link_op_PKG;



package body cnr_link_op_PKG is


 procedure cnr_link_processing_delay is
 begin
   delay duration(constants_PKG.mean_cnr_link_processing_delay);--temporary;use mean & deviation to
                                        --calculate delay(see constants_PKG)
 end cnr_link_processing_delay;

 procedure cnr_link_transmission_prep_delay is
 begin
   delay duration(constants_PKG.mean_cnr_link_transmission_prep_delay);--temporary;use mean &
                                                                          --deviation to calculate
                                                                          --delay(see constants_PKG)
 end cnr_link_transmission_prep_delay;



 procedure cnr_link_op (fire_cmd3_str : in target_data_PKG.target_data;
               fire_cmd4_str : out target_data_PKG.target_data) is

          package flt_io is new float_io(float);
    use flt_io;
          my_random : float := 0.0;

begin
          -- below is a way to simulate friction,

          my_random := global_random_PKG.ftn_global_random;

          if my_random > constants_PKG.crash_rate then
                    fire_cmd4_str := fire_cmd3_str;
                    fire_cmd4_str.good_xmission :=  false;  -- set flag
                    --put_line("Now in procedure 'cnr_link_op' ... crash occured");
          else
                    fire_cmd4_str := fire_cmd3_str;
          --put_line("Now in procedure 'cnr_link_op'");
          end if;
```

```
--THIS SIMULATES THE LATENCY FROM THE CTOC TO SHOOTER
          delay 3.0;

--THESE ARE PLACEHOLDERS; USE/MODIFY AS NEEDED
   --cnr_link_processing_delay;
   --cnr_link_transmission_prep_delay;


 end cnr_link_op;

end cnr_link_op_PKG;
```

atacms.constants.a

```
-- file: atacms.constants.a
--
-- Authors:  Maj George Whitbeck and LCDR David Angrisani
-- Date:     6 Aug 96
-- Project:  Thesis - A CAPS Prototype of the ATACMS C3 System
-- Purpose:  To hold constants  that we deemed necessary.


--To create a library: 'a.mklib -i' then choose '1'
--to clear a library:  'a.rmlib'
-- compile with... ada atacms.constants.a


with text_io;
use text_io;
with my_unit_PKG;

package constants_PKG is

 -- to allow i/o for booleans
 package bool_io is NEW text_io.Enumeration_io(Enum => boolean);

 -- to allow i/o for integers
 package int_io is new integer_io (integer);


 subtype delay_type is float range 0.0..float'last;
  mean_jstars_operator_delay:  delay_type:= 5.0;
  mean_jstars_processing_delay:  delay_type:= 5.0;
  mean_jstars_transmission_prep_delay:  delay_type:= 5.0;
  mean_scdl_link_processing_delay:  delay_type:= 5.0;
  mean_scdl_link_transmission_prep_delay:  delay_type:= 5.0;
  mean_grnd_stat_mod_operator_delay:  delay_type:= 5.0;
  mean_grnd_stat_mod_processing_delay:  delay_type:= 5.0;
  mean_grnd_stat_mod_transmission_prep_delay:  delay_type:= 5.0;
  mean_lan1_link_processing_delay:  delay_type:= 5.0;
  mean_lan1_link_transmission_prep_delay:  delay_type:= 5.0;
  mean_asas_operator_delay:  delay_type:= 5.0;
  mean_asas_processing_delay:  delay_type:= 5.0;
  mean_asas_transmission_prep_delay:  delay_type:= 5.0;
  mean_lan2_link_processing_delay:  delay_type:= 5.0;
  mean_lan2_link_transmission_prep_delay:  delay_type:= 5.0;
  mean_ctoc_operator_delay:  delay_type:= 5.0;
  mean_ctoc_processing_delay:  delay_type:= 5.0;
  mean_ctoc_transmission_prep_delay:  delay_type:= 5.0;
  mean_cnr_link_processing_delay:  delay_type:= 5.0;
  mean_cnr_link_transmission_prep_delay:  delay_type:= 5.0;
  mean_shooter_operator_delay:  delay_type:= 5.0;
  mean_shooter_processing_delay:  delay_type:= 5.0;
  mean_shooter_transmission_prep_delay:  delay_type:= 5.0;


 subtype deviation is float range 0.0..float'last;
  jstars_operator_deviation:  deviation:= 5.0;
  jstars_processing_deviation:  deviation:= 5.0;
  jstars_transmission_prep_deviation:  deviation:= 5.0;
  scdl_link_processing_deviation:  deviation:= 5.0;
  scdl_link_transmission_prep_deviation:  deviation:= 5.0;
  grnd_stat_mod_operator_deviation:  deviation:= 5.0;
  grnd_stat_mod_processing_deviation:  deviation:= 5.0;
  grnd_stat_mod_transmission_prep_deviation:  deviation:= 5.0;
  lan1_link_processing_deviation:  deviation:= 5.0;
  lan1_link_transmission_prep_deviation:  deviation:= 5.0;
  asas_operator_deviation:  deviation:= 5.0;
  asas_processing_deviation:  deviation:= 5.0;
  asas_transmission_prep_deviation:  deviation:= 5.0;
```

```
  lan2_link_processing_deviation: deviation:= 5.0;
  lan2_link_transmission_prep_deviation: deviation:= 5.0;
  ctoc_operator_deviation: deviation:= 5.0;
  ctoc_processing_deviation: deviation:= 5.0;
  ctoc_transmission_prep_deviation: deviation:= 5.0;
  cnr_link_processing_deviation: deviation:= 5.0;
  cnr_link_transmission_prep_deviation: deviation:= 5.0;
  shooter_operator_deviation: deviation:= 5.0;
  shooter_processing_deviation: deviation:= 5.0;
  shooter_transmission_prep_deviation: deviation:= 5.0;


 min_array   : constant integer := 100;  -- the min array index
 max_array   : constant integer := 110;  -- the max array index
 thousand    : constant integer := 1000; -- represents magic number of 1000
 crash_rate  : constant float   := 0.9;  -- the percent of successful transmissions
 max_new_tgts : constant integer := 8;   -- the number of new target detections each cycle

end constants_PKG;
```

atacms.ctoc_op.a

```
-- filename:  atacms.ctoc_op.a
--
-- Authors:  Maj George Whitbeck and LCDR David Angrisani
-- Date:     6 Aug 96;modofied 04 Sep 96
-- Project:  Thesis - A CAPS Prototype of the ATACMS C3 System
-- Purpose:    This package simulates the operations of the CTOC.
--   Their function in this abstraction is nil.  They simply send
--   off the fire mission to the shooter
--
-- compile..  ada atacms.ctoc_op.a


with text_io; use text_io;
with target_data_PKG;
with constants_PKG;
package ctoc_op_PKG is

 procedure ctoc_op (fire_cmd2_str :  target_data_PKG.target_data;
              fire_cmd3_str : out target_data_PKG.target_data);

end ctoc_op_PKG;



package body ctoc_op_PKG is


 procedure ctoc_operator_delay is
 begin
   delay duration(constants_PKG.mean_ctoc_operator_delay); --temporary;use mean & deviation
                                                            --to calculate delay
                                                            --(see constants_PKG)
 end ctoc_operator_delay;

 procedure ctoc_processing_delay is
 begin
   delay duration(constants_PKG.mean_ctoc_processing_delay);--temporary;use mean & deviation
                                                            --to calculate delay(see
                                                            --constants_PKG)
 end ctoc_processing_delay;

 procedure ctoc_transmission_prep_delay is
 begin
   delay duration(constants_PKG.mean_ctoc_transmission_prep_delay);--temporary;use mean &
                                                                   --deviation to calculate
                                                                   --delay(see constants_PKG)
 end ctoc_transmission_prep_delay;

 procedure ctoc_error is
 begin
   null;                              --insert simulated errors
 end ctoc_error;


 procedure ctoc_op (fire_cmd2_str :  target_data_PKG.target_data;
              fire_cmd3_str : out target_data_PKG.target_data) is


 begin

     fire_cmd3_str := fire_cmd2_str;
     --Put_line("Now in procedure `ctoc_op'");
```

```
--THESE ARE PLACEHOLDERS; USE/MODIFY AS NECESSARY
   --ctoc_operator_delay;
   --ctoc_processing_delay;
   --ctoc_transmission_prep_delay;
   --ctoc_error;


 end ctoc_op;

end ctoc_op_PKG;
```

atacms.event_task.a

```
-- filename: atacms.event_task.a
--
-- Authors:  Maj George Whitbeck and LCDR David Angrisani
-- Date:     6 Aug 96
-- Project:  Thesis - A CAPS Prototype of the ATACMS C3 System
-- Purpose:  The wrapper task to provide mutual exclusion
--           for calls from the prototype to TAE.

with target_data_PKG; --######## we added this
with text_io; use text_io;

package atacms_input_event_monitor_task_pkg is
 task atacms_input_event_monitor_task is
   entry input_event_monitor_entry;
   entry cmds_out_entry(gui_out_str: target_data_PKG.target_data);
 end atacms_input_event_monitor_task;
end atacms_input_event_monitor_task_pkg;

 with panel_gui_out;
 with generated_tae_input_event_monitor_pkg;
package body atacms_input_event_monitor_task_pkg is
 task body atacms_input_event_monitor_task is
 begin
  loop
   select
    accept input_event_monitor_entry do
     --put_line("@@@@@@@@ in 'event_task.a', now accepting an INPUT");
     generated_tae_input_event_monitor_pkg.generated_tae_input_event_monitor;
     --put_line("@@@@@@@@ in 'event_task.a', now returning from an INPUT");

    end input_event_monitor_entry;
   or
    accept cmds_out_entry(gui_out_str: target_data_PKG.target_data) do
     --put_line("%%%%%%%% in 'event_task.a', now accepting an OUTPUT");
     panel_gui_out.cmds_out(gui_out_str);
     --put_line("%%%%%%%% in 'event_task.a', now returning from an OUTPUT");

    end cmds_out_entry;
   end select;
  end loop;
 end atacms_input_event_monitor_task;
end atacms_input_event_monitor_task_pkg;
```

atacms.global_b.a

```
-- filename: atacms.global_b.a
-- Authors: Maj George Whitbeck and LCDR David Angrisani
-- Date:    6 Aug 96
-- Project: Thesis - A CAPS Prototype of the ATACMS C3 System
-- Purpose: This TAE generated file is handy for adding a quit button to CAPS prototypes


-- *** TAE Plus Code Generator version V5.3 [Merge Token: DO NOT DELETE.]
-- *** File:      global_b.a
-- *** Generated: Jul 30 13:48:56 1996
-- ************************************************************************
-- *
-- *   Global                  -- Package BODY
-- *
-- ************************************************************************

package body Global is

--| NOTES: (none)
--|
--| REGENERATED:
--| This file is generated only once. Therefore, you may modify it without
--| impacting automatic code merge.
--|
--| CHANGE LOG:
--| 30-Jul-96   TAE    Generated

 Is_Application_Done : Boolean := FALSE;

 -- ....................................................................
 -- .
 -- .  Application_Done          -- Subprogram BODY
 -- .
 -- ....................................................................

 function Application_Done
  return Boolean is

 --| NOTES: (none)

 begin -- Application_Done

  return Is_Application_Done;

 end Application_Done;

 -- ....................................................................
 -- .
 -- .  Set_Application_Done      -- Subprogram BODY
 -- .
 -- ....................................................................

 procedure Set_Application_Done is

 --| NOTES: (none)

 begin -- Set_Application_Done

  Is_Application_Done := TRUE;

 end Set_Application_Done;

end Global;
```

atacms.global_random.a

```
-- filename "atacms.global_random.a"
--
-- Authors: Maj George Whitbeck and LCDR David Angrisani
-- Date:    6 Aug 96
-- Project: Thesis - A CAPS Prototype of the ATACMS C3 System
-- Purpose: A random number generator.  A call to this function returns a
--          real number from 0.0 to 1.0


-- original code from Prof Shing, modified by Maj Whitbeck
-- 6/12/96

-- to compile "ada filename" ... "ada global_random.a"


with CALENDAR;
use  CALENDAR;

package global_random_PKG is

 subtype Global_Random_Number is Float range 0.0..1.0;

 function ftn_global_random return Global_Random_Number;
 --returns a random number between 0.0 and 1.0

end global_random_PKG;


package body global_random_PKG is

 Seed_Is_Not_Initialized : Boolean := True;
 Max_Seed_Value          : constant Natural := 3**9;
 Seed                    : Natural := 0;
 Does_Not_Matter         : constant Natural := 5**7;

 procedure Initialize_Seed is
 --initializes seed value by using clock time, so that at each run of the
 --package, a different set of numbers are created
 begin
  while ( Seed mod 2 ) = 0 loop
   Seed := ( Natural (Seconds(Clock)) ) mod Max_Seed_Value;
  end loop;
 end Initialize_Seed;

 function ftn_global_random return  Global_Random_Number is
 begin
  if Seed_Is_Not_Initialized then Initialize_Seed;
    Seed_Is_Not_Initialized := False;
  end if;
  Seed := ( Seed * Does_Not_Matter ) mod Max_Seed_Value;
  return ( Float( Seed ) / Float( Max_Seed_Value ) );
 end ftn_global_random;

end global_random_PKG;
```

atacms.global_s.a

```
-- Authors:  Maj George Whitbeck and LCDR David Angrisani
-- Date:    6 Aug 96
-- Project:  Thesis - A CAPS Prototype of the ATACMS C3 System
-- Purpose:


-- *** TAE Plus Code Generator version V5.3 [Merge Token: DO NOT DELETE.]
-- *** File:       global_s.a
-- *** Generated:  Jul 30 13:48:56 1996
-- ****************************************************************************
-- *
-- *   Global                    -- Package SPEC
-- *
-- ****************************************************************************

with X_Windows;
with Text_IO;
with TAE;

package Global is

--| PURPOSE:
--| This package is automatically "with"ed in to each panel package body.
--| You can insert global variables here.
--|
--| INITIALIZATION EXCEPTIONS: (none)
--|
--| NOTES: (none)
--|
--| REGENERATED:
--| This file is generated only once.  Therefore, you may modify it without
--| impacting automatic code merge.
--|
--| CHANGE LOG:
--| 30-Jul-96   TAE     Generated

 package Taefloat_IO is new Text_IO.Float_IO (TAE.Taefloat);

 Default_Display_Id : X_Windows.Display;

-- ......................................................................
-- .
-- .   Application_Done            -- Subprogram SPEC
-- .
-- ......................................................................

function Application_Done
  return Boolean;

--| PURPOSE:
--| This function returns true if a "quit" event handler has called
--| Set_Application_Done, otherwise it returns false.
--|
--| EXCEPTIONS: (none)
--|
--| NOTES: (none)


-- ......................................................................
-- .
-- .   Set_Application_Done        -- Subprogram SPEC
-- .
-- ......................................................................
```

```
procedure Set_Application_Done;

--| PURPOSE:
--| This procedure can be used by an event handler, typically a "quit"
--| button, to signal the end of the application.
--|
--| EXCEPTIONS: (none)
--|
--| NOTES: (none)

end Global;
```

atacms.grnd_stat_mod_array.a

```
-- filename: atacms.grnd_stat_mod_array.a
--
-- Authors:  Maj George Whitbeck and LCDR David Angrisani
-- Date:     6 Aug 96
-- Project:  Thesis - A CAPS Prototype of the ATACMS C3 System
-- Purpose:

-- note naming convention here with filename, package name, and type name!!!!!
-- to just compile type... ada atacms.grnd_stat_mod_array.a


with text_io;
use text_io;
with constants_PKG;

package grnd_stat_mod_array_PKG is

  --THIS SECTION CONTAINS grnd_stat_mod_array, which is the type for the
  -- output stream for grnd_stat_mod_op and lan1_link_op and the
  -- input stream for lan1_link_op and asas_op

 -- to create legal entries for the UTM coordinate system
  subtype east_coord is integer range 0..999_999;
  subtype north_coord is integer range 0..9_999_999;
  subtype alt_coord is integer range -1000..10_000;



   type grnd_stat_mod_array_record is record
     target_class: integer:=0;
     easting : east_coord := 0;
     northing : north_coord := 0;
     alt : alt_coord := 0;
     tgt_num : natural := 0;
     good_xmission : boolean := true;
     status : natural := 0;

   end record;

   type grnd_stat_mod_array is array(constants_PKG.min_array..constants_PKG.max_array,
                           constants_PKG.min_array..constants_PKG.max_array)
                           of grnd_stat_mod_array_record;

end grnd_stat_mod_array_PKG;
```

atacms.grnd_stat_mod_op.a

```
-- filename: atacms.grnd_stat_mod_op.a
-- Authors: Maj George Whitbeck and LCDR David Angrisani
-- Created 05 Jun 96, mod 7/29/96; MOD 04 SEP 96
-- Project: Thesis - A CAPS Prototype of the ATACMS C3 System

-- Purpose:    This package simulates the operations of the
--    ground station. In the ground station, the video is analyzed and target locations
--    are determined. In this prototype, each target coming in is given a simulated
--    location (see below). This station also assigns a unique target number to each target.

--    compile... ada atacms.grnd_stat_mod_op.a


with text_io; use text_io;
with grnd_stat_mod_array_PKG;
with jstars_array_PKG;
with constants_PKG;
with global_random_PKG;

package grnd_stat_mod_op_PKG is

 procedure grnd_stat_mod_op
          (target_array2_str : jstars_array_PKG.jstars_array;
           target_array3_str : out grnd_stat_mod_array_PKG.grnd_stat_mod_array);

end grnd_stat_mod_op_PKG;


package body grnd_stat_mod_op_PKG is


 -- This array has a memory due to its position in this package
 -- Used to store and manipulate similated info before dumping its
 -- contents into the OUT parameter.
 LV_target_array : grnd_stat_mod_array_PKG.grnd_stat_mod_array;

 -- this will record a unique serial number for each target identified
 LV_tgt_num : natural := 0;

 procedure grnd_stat_mod_operator_delay is
 begin
   delay duration(constants_PKG.mean_grnd_stat_mod_operator_delay); --temporary;use mean & deviation
                                                                     --to calculate delay
                                                                     --(see constants_PKG)
 end grnd_stat_mod_operator_delay;

 procedure grnd_stat_mod_processing_delay is
 begin
   delay duration(constants_PKG.mean_grnd_stat_mod_processing_delay);--temporary;use mean & deviation
                                                                     --to calculate delay(see
                                                                     --constants_PKG)
 end grnd_stat_mod_processing_delay;

 procedure grnd_stat_mod_transmission_prep_delay is
 begin
   delay duration(constants_PKG.mean_grnd_stat_mod_transmission_prep_delay);--temporary;use mean &
                                                                            --deviation to calculate
                                                                            --delay(see constants_PKG)
 end grnd_stat_mod_transmission_prep_delay;
```

```
procedure grnd_stat_mod_error is
begin
  null;                                 --insert simulated errors
end grnd_stat_mod_error;


procedure grnd_stat_mod_op
        (target_array2_str : jstars_array_PKG.jstars_array;
         target_array3_str : out grnd_stat_mod_array_PKG.grnd_stat_mod_array) is

          package flt_io is new float_io(float);
   use flt_io;
   my_random : float := 0.0;
          easting  : natural := 0;
          northing : natural := 0;
          alt      : natural := 0;

begin
   --Put_line("Now in procedure `grnd_stat_mod_op'");

   --put("Good xmission so far? ... ");
   --constants_PKG.bool_io.put(target_array2_str(constants_PKG.min_array,
   --             constants_PKG.min_array).good_xmission); new_line;



         -- Step thru the video (array of records)
         for row_x in constants_PKG.min_array..constants_PKG.max_array loop
           for column_y in constants_PKG.min_array..constants_PKG.max_array loop

             -- If a new target exists, determine its location.
             -- It's new only if there is a reflection there and you
             --  haven't seen it before.
      if (target_array2_str(row_x,column_y).target_class /= 0)  AND
                   (LV_target_array(row_x,column_y).tgt_num = 0)  then

                   my_random := global_random_PKG.ftn_global_random;
       -- now ensure the correct bounds for the random number
                   --  to make the grid coordinate a realistic number
       while ((my_random < 0.1) or (my_random > 0.9)) loop
                             my_random := global_random_PKG.ftn_global_random;
                   end loop;

                   -- Our way of assigning grid coordinates to the newly acquired target.
                   easting := integer(100000.0 * my_random);
                   northing := 100000 - easting;
                   alt := integer(100.0 * my_random);

                   -- now transfer to information to local video
        LV_target_array(row_x,column_y).target_class :=
          target_array2_str(row_x,column_y).target_class;
                   LV_target_array(row_x,column_y).easting := easting;
                   LV_target_array(row_x,column_y).northing := northing;
                   LV_target_array(row_x,column_y).alt := alt;
                   LV_tgt_num := LV_tgt_num + 1;  -- create a new target number***
                   LV_target_array(row_x,column_y).tgt_num := LV_tgt_num;


          else
                   -- no target here, just pass on info
                   LV_target_array(row_x,column_y).target_class :=
         target_array2_str(row_x,column_y).target_class;
          end if;

         end loop;
       end loop;
```

```
--THESE ARE PLACEHOLDERS; USE/MODIFY AS NECESSARY
      --grnd_stat_mod_operator_delay;
      --grnd_stat_mod_processing_delay;
      --grnd_stat_mod_transmission_prep_delay;
      --grnd_stat_mod_error;


            -- pass on the transmission status, it does not get affected here
            LV_target_array(constants_PKG.min_array,constants_PKG.min_array).good_xmission :=
              target_array2_str(constants_PKG.min_array,constants_PKG.min_array).good_xmission;

            -- now do a direct copy to the OUT parameter
      target_array3_str := LV_target_array;

  end grnd_stat_mod_op;

end grnd_stat_mod_op_PKG;


--constants_PKG.int_io.put(easting,0);
--put(" & ");
--constants_PKG.int_io.put(northing,0);
--put(" & ");
--constants_PKG.int_io.put(alt,0);
--put(" & tgt_num is ");
--constants_PKG.int_io.put(LV_tgt_num,0);
-- new_line(2);


--      Put("Now in procedure `grnd_stat_mod_op_', just got a ");
--      constants_PKG.int_io.put(target_array2_str(row_x,column_y).target_class,0);
--      new_line;

--put("*********** row and col are ");
-- constants_PKG.int_io.put(row_x,0);
--put(" & ");
--constants_PKG.int_io.put(column_y,0); new_line;

--constants_PKG.int_io.put(easting,0);
--put(" & ");
--constants_PKG.int_io.put(northing,0);
--put(" & ");
--constants_PKG.int_io.put(alt,0);
--new_line;

--    put("A random number is ");
--    put(my_random,1,3,0); new_line;

     --target_array3_str.easting := target_array2_str(100,100).target_class;
```

atacms.gui_input_event_monitor.a

```
-- filename: atacms.gui_input_event_monitor.a
--
-- Authors:  Maj George Whitbeck and LCDR David Angrisani
-- Date:     6 Aug 96
-- Project:  Thesis - A CAPS Prototype of the ATACMS C3 System
-- Purpose:  This comes from Dr Berzin's design for the multi-file approach to caps.
-- "gui_input_event_monitor" is a child bubble with no streams.  It is embedded in the
-- parent "gui_in".  It's only purpose is to request a rendezvous with the event monitor
-- in file "atacms.event_task.a"

with text_io; use text_io;

package gui_input_event_monitor_pkg is
  procedure gui_input_event_monitor;
end gui_input_event_monitor_pkg;

  with atacms_input_event_monitor_task_pkg;
  use atacms_input_event_monitor_task_pkg;
package body gui_input_event_monitor_pkg is
  procedure gui_input_event_monitor is
  begin
    --put_line("about to request a rendezvous... in gui_input_event_monitor.a ");
    atacms_input_event_monitor_task.input_event_monitor_entry;
  end gui_input_event_monitor;
end gui_input_event_monitor_pkg;
```

atacms.jstars_array.a

```
-- filename: atacms.jstars_array.a
--
-- Authors:  Maj George Whitbeck and LCDR David Angrisani
-- Date:     6 Aug 96
-- Project:  Thesis - A CAPS Prototype of the ATACMS C3 System
-- Purpose:  This file is where we declare an array to simulate jstars output video
-- that we use as a stream.
--

-- to just compile type... ada atacms.jstars_array.a


with text_io;
use text_io;
with constants_PKG;


package jstars_array_PKG is

--THIS SECTION CONTAINS jstars_array, which is the type for the
-- output stream for jstars_op.a, the input/output stream for scdl_link_op.a,
-- and the input stream for grnd_stat_mod_op.a

-- to create legal entries for the UTM coordinate system
 subtype east_coord is integer range 0..999_999;
 subtype north_coord is integer range 0..9_999_999;
 subtype alt_coord is integer range -1000..10_000;



 type jstars_array_record is record
  target_class: integer:=0;
  easting : east_coord := 0;
  northing : north_coord := 0;
  tgt_num : natural := 0;
  good_xmission : boolean := true;
  status : natural := 0;

 end record;

 type jstars_array is array(constants_PKG.min_array..constants_PKG.max_array,
                  constants_PKG.min_array..constants_PKG.max_array)
                  of jstars_array_record;


end jstars_array_PKG;
```

atacms.jstars_op.a

```
-- filename:  atacms.jstars_op.a
--
-- Authors:  Maj George Whitbeck and LCDR David Angrisani
-- Date:    6 Aug 96l; modified 04 Sep 96
-- Project:  Thesis - A CAPS Prototype of the ATACMS C3 System

-- Purpose:    This package simulates the operations of the JSTARS Platform
--  It collects emmisions, creates the digital video and transmits it to the ground station


with text_io; use text_io;
with target_emitter_array_PKG;
with jstars_array_PKG;
with constants_PKG;
with my_unit_PKG;


package jstars_op_PKG is

 procedure jstars_op (emission_str : target_emitter_array_PKG.target_emitter_array;
              gui_in_str : my_unit_PKG.my_unit;
              target_array1_str : out jstars_array_PKG.jstars_array);

 --procedure jstars_operator_delay (mean_jstars_operator_delay: in operator_delay);

end jstars_op_PKG;



package body jstars_op_PKG is


 procedure jstars_operator_delay is
 begin
   delay duration(constants_PKG.mean_jstars_operator_delay); --temporary;use mean & deviation to
                                    --calculate delay(see constants_PKG)
 end jstars_operator_delay;

 procedure jstars_processing_delay is
 begin
   delay duration(constants_PKG.mean_jstars_processing_delay);--temporary;use mean & deviation to
                                    --calculate delay(see constants_PKG)
 end jstars_processing_delay;

 procedure jstars_transmission_prep_delay is
 begin
   delay duration(constants_PKG.mean_jstars_transmission_prep_delay);--temporary;use mean & deviation
                                                        --to calculate delay(see
                                                                  --constants_PKG)
 end jstars_transmission_prep_delay;

 procedure jstars_error is
 begin
   null;                                --insert simulated errors
 end jstars_error;


 procedure jstars_op (emission_str : target_emitter_array_PKG.target_emitter_array;
              gui_in_str : my_unit_PKG.my_unit;
              target_array1_str : out jstars_array_PKG.jstars_array) is
```

```
begin

  put_line("JSTARS receiving radar reflections, resolving duplications, and transmitting video...");
  for row_x in  constants_PKG.min_array..constants_PKG.max_array loop

   for column_y in constants_PKG.min_array..constants_PKG.max_array loop

    -- one for one copy of in to out in this abstraction
    target_array1_str(row_x,column_y).target_class:=
       emission_str(row_x,column_y).target_class;

   end loop;
  end loop;

--THESE ARE PLACEHOLDERS; USE/MODIFY AS NEEDED
  --jstars_operator_delay;
  --jstars_processing_delay;
  --jstars_transmission_prep_delay;
  --jstars_error;

 end jstars_op;

end jstars_op_PKG;
```

atacms.lan1_link_op.a

```
-- filename: atacms.lan1_link_op.a
--
-- Authors:  Maj George Whitbeck and LCDR David Angrisani
-- Date:     6 Aug 96;modified 04 Sep 96
-- Project:  Thesis - A CAPS Prototype of the ATACMS C3 System
-- Purpose:

-- this comm link will pass on 90% of traffic and corrupt 10%
--
-- to compile  type... ada atacms.lan1_link_op.a

with text_io; use text_io;
with global_random_PKG;
with grnd_stat_mod_array_PKG;
with constants_PKG;

package lan1_link_op_PKG is

 procedure lan1_link_op
        (target_array3_str : in grnd_stat_mod_array_PKG.grnd_stat_mod_array;
         target_array4_str : out grnd_stat_mod_array_PKG.grnd_stat_mod_array );

end lan1_link_op_PKG;



package body lan1_link_op_PKG is

 procedure lan1_link_processing_delay is
 begin

   delay duration(constants_PKG.mean_lan1_link_processing_delay);--temporary;use mean & deviation to
                                                    --calculate delay(see constants_PKG)
 end lan1_link_processing_delay;


 procedure lan1_link_transmission_prep_delay is
 begin
   delay duration(constants_PKG.mean_lan1_link_transmission_prep_delay);--temporary;use mean &
                                                                         --deviation to calculate
                                                                         --delay(see constants_PKG)
 end lan1_link_transmission_prep_delay;



 procedure lan1_link_op
        (target_array3_str : in grnd_stat_mod_array_PKG.grnd_stat_mod_array;
         target_array4_str : out grnd_stat_mod_array_PKG.grnd_stat_mod_array) is

            package flt_io is new float_io(float);
    use flt_io;
            my_random : float := 0.0;

 begin

   --put_line("Now in procedure 'lan1_link_op'");

   --put("Good xmission so far? ... ");
   --constants_PKG.bool_io.put(target_array3_str(constants_PKG.min_array,
   --                constants_PKG.min_array).good_xmission); new_line;


   -- a direct copy of the array of records
   target_array4_str := target_array3_str;

   -- below is a way to simulate friction
```

```
   my_random := global_random_PKG.ftn_global_random;

   if my_random > constants_PKG.crash_rate then     -- simulate crash,put flag in first position

          target_array4_str(constants_PKG.min_array,constants_PKG.min_array).good_xmission
        := false;
          --put_line("Still in procedure 'lan1_link_op' ... crash occured");

   end if;

--THIS SIMULATES THE LATENCY FROM THE GROUND STATION MODULE TO ASAS
          delay 1.0;


--THESE ARE PLACEHOLDERS; USE MODIFY AS NEEDED
   --lan1_link_processing_delay;
   --lan1_link_transmission_prep_delay;

 end lan1_link_op;

end lan1_link_op_PKG;
```

atacms.lan2_link_op.a

```
-- filename: atacms.lan2_link_op.a
--
-- Authors:  Maj George Whitbeck and LCDR David Angrisani
-- Date:     6 Aug 96;mod 04 Sep 96
-- Project:  Thesis - A CAPS Prototype of the ATACMS C3 System
-- Purpose:  This comm link will now just pass on the data and corrupt 10% of it.
--
-- to compile... ada atacms.lan2_link_op.a

with text_io; use text_io;
with target_data_PKG;  -- to instantiate my record structure
with global_random_PKG;  -- a function to get a random number
with constants_PKG;

package lan2_link_op_PKG is
  procedure lan2_link_op (fire_cmd1_str : in target_data_PKG.target_data;
                fire_cmd2_str : out target_data_PKG.target_data );
end lan2_link_op_PKG;



package body lan2_link_op_PKG is


 procedure lan2_link_processing_delay is
 begin
   delay duration(constants_PKG.mean_lan2_link_processing_delay);--temporary;use mean & deviation to
                                                   --calculate delay(see constants_PKG)
 end lan2_link_processing_delay;

 procedure lan2_link_transmission_prep_delay is
 begin
   delay duration(constants_PKG.mean_lan2_link_transmission_prep_delay);--temporary;use mean &
                                                                                   --deviation to calculate
                                                                                   --delay(see constants_PKG)
 end lan2_link_transmission_prep_delay;



 procedure lan2_link_op (fire_cmd1_str : in target_data_PKG.target_data;
                fire_cmd2_str : out target_data_PKG.target_data) is

        package flt_io is new float_io(float);
    use flt_io;
        my_random : float := 0.0;

begin

        --put_line("Now in procedure 'lan2_link_op'");

        -- a direct copy of record
        fire_cmd2_str := fire_cmd1_str;

        -- below is a way to simulate friction
        -- 95% of the time, everything will be fine.
        my_random := global_random_PKG.fn_global_random;

        if my_random > constants_PKG.crash_rate then    -- set flag
                fire_cmd2_str.good_xmission := false;
                --put_line("Still in procedure 'lan2_link_op' ... crash occured");
        end if;
```

```
--THIS SIMULATES THE LATENCY FROM THE ASAS TO CTOC
          delay 1.0;

--THESE ARE PLACEHOLDERS; USE/MODIFY AS NECESSARY
   --lan2_link_processing_delay;
   --lan2_link_transmission_prep_delay;


 end lan2_link_op;

end lan2_link_op_PKG;
```

atacms.my_unit.a

```
-- file: atacms.my_unit.a
--
-- Authors:  Maj George Whitbeck and LCDR David Angrisani
-- Date:     6 Aug 96
-- Project:  Thesis - A CAPS Prototype of the ATACMS C3 System
-- Purpose:  This file specifies a type which closely matches strings read from a TAE
--           input panel.
-- compile with... ada atacms.my_unit.a

with text_IO;

package my_unit_PKG is


   type my_unit is (Pause, Go, Add_Target, Quit);

   package my_unit_IO is
        new text_IO.Enumeration_IO(Enum => my_unit);

end my_unit_PKG;
```

atacms.pan_gui_in_b.a

```
-- filename: atacms.pan_gui_in_b.a
--
-- Authors:  Maj George Whitbeck and LCDR David Angrisani
-- Date:    6 Aug 96
-- Project:  Thesis - A CAPS Prototype of the ATACMS C3 System
-- Purpose:  See comments between merge notes below

-- *** TAE Plus Code Generator version V5.3 [Merge Token: DO NOT DELETE.]
-- *** File:      pan_gui_in_b.a
-- *** Generated:  Jul 30 13:48:56 1996
-- ************************************************************************
-- *
-- *   Panel_gui_in              -- Package BODY
-- *
-- ************************************************************************
with TAE; use TAE;
with Text_IO;
with Global;

-- One "with" statement for each connected panel.

-- MERGE NOTE: Add additional "with"s BELOW this line.

-- ***************************
-- modified by GSW on 7/30/96
-- file: "atacms.pan_gui_in_b.a"
-- my mods are noted by "--####" below
-- this files has an if statement added in the event handler and a few
-- with statements

--########
with my_unit_PKG;       -- provides the type for the input event
with choose_inputs_PKG;

-- MERGE NOTE: Add additional "with"s ABOVE this line.

package body Panel_gui_in is

--| NOTES:
--| For each parameter that you have defined to be "event-generating" in
--| this panel, there is an event handler procedure below.  Each handler
--| has a name that is a concatenation of the parameter name and "_Event".
--| Add application-dependent logic to each event handler.  (As generated
--| by the WorkBench, each event handler simply logs the occurrence of
--| the event.)
--|
--| For best automatic code merging results, you should put as many
--| modifications as possible between the lines of the MERGE NOTE comments.
--| Modifications outside the MERGE NOTEs will often merge correctly, but
--| must sometimes be merged by hand.  If the modifications cannot be
--| automatically merged, a reject file (*.rej) will be generated which
--| will contain your modifications.
--|
--| REGENERATED:
--| The following WorkBench operations will cause regeneration of this file:
--|    The panel's name is changed (not title)
--| For panel:  gui_in
--|
--| The following WorkBench operations will also cause regeneration:
--|    An item is deleted
--|    A new item is added to this panel
--|    An item's name is changed (not title)
--|    An item's data type is changed
--|    An item's generates events flag is changed
--|    An item's valids changed (if item is type string and connected)
--|    An item's connection information changed
```

```
--| For the panel items:
--|   choose_inputs,  header
--|
--| CHANGE LOG:
--| MERGE NOTE: Add Change Log entries BELOW this line.
--| 30-Jul-96   TAE     Generated
--| MERGE NOTE: Add Change Log entries ABOVE this line.

 -- MERGE NOTE: Add additional code BELOW this line.
 -- MERGE NOTE: Add additional code ABOVE this line.

 -- ......................................................................
 -- .
 -- .  Initialize_Panel             -- Subprogram BODY
 -- .
 -- ......................................................................

 procedure Initialize_Panel
  ( Collection_Read
    : in TAE.Tae_Co.Collection_Ptr ) is

 --| NOTES: (none)

  -- MERGE NOTE: Add declarations for Initialize_Panel BELOW this line.
  -- MERGE NOTE: Add declarations for Initialize_Panel ABOVE this line.

 begin -- Initialize_Panel

  Info := new TAE.Tae_Wpt.Event_Context;
  Info.Collection := Collection_Read;
  TAE.Tae_Co.Co_Find (Info.Collection, "gui_in_v", Info.View);
  TAE.Tae_Co.Co_Find (Info.Collection, "gui_in_t", Info.Target);

  -- MERGE NOTE: Add code for Initialize_Panel BELOW this line.
  -- MERGE NOTE: Add code for Initialize_Panel ABOVE this line.

 exception

  when TAE.UNINITIALIZED_PTR =>
   Text_IO.Put_Line ("Panel_gui_in.Initialize_Panel:  "
    & "Collection_Read not initialized.");
   raise;

  when TAE.Tae_Co.NO_SUCH_MEMBER =>
   Text_IO.Put_Line ("Panel_gui_in.Initialize_Panel:  "
    & "(View or Target) not in Collection.");
   raise;

 end Initialize_Panel;
```

```
-- ..................................................................
-- .
-- .  Create_Panel                 -- Subprogram BODY
-- .
-- ..................................................................

procedure Create_Panel
 ( Panel_State
    : in TAE.Tae_Wpt.Wpt_Flags
     := TAE.Tae_Wpt.WPT_PREFERRED;

   Relative_Window
    : in X_Windows.Window
     := X_Windows.Null_Window ) is

--| NOTES: (none)

 -- MERGE NOTE: Add declarations for Create_Panel BELOW this line.
 -- MERGE NOTE: Add declarations for Create_Panel ABOVE this line.

begin -- Create_Panel

 if Info.Panel_Id = Tae.Null_Panel_Id then
  TAE.Tae_Wpt.Wpt_NewPanel
   ( Display_Id        => Global.Default_Display_Id,
     Data_Vm           => Info.Target,
     View_Vm           => Info.View,
     Relative_Window   => Relative_Window,
     User_Context      => Info,
     Flags             => Panel_State,
     Panel_Id          => Info.Panel_Id );
 else
  Text_IO.Put_Line ("Panel (gui_in) is already displayed.");
 end if;

 -- MERGE NOTE: Add code for Create_Panel BELOW this line.
 -- MERGE NOTE: Add code for Create_Panel ABOVE this line.

exception

 when TAE.UNINITIALIZED_PTR =>
  Text_IO.Put_Line ("Panel_gui_in.Create_Panel: "
   & "Panel was not initialized prior to creation.");
  raise;

 when TAE.TAE_FAIL =>
  Text_IO.Put_Line ("Panel_gui_in.Create_Panel: "
   & "Panel could not be created.");
  raise;

end Create_Panel;
```

```
-- ......................................................................
-- .
-- .  Connect_Panel            -- Subprogram BODY
-- .
-- ......................................................................

procedure Connect_Panel
 ( Panel_State
   : in TAE.Tae_Wpt.Wpt_Flags
     := TAE.Tae_Wpt.WPT_PREFERRED;

   Relative_Window
    : in X_Windows.Window
     := X_Windows.Null_Window ) is

--| NOTES: (none)

 -- MERGE NOTE: Add declarations for Connect_Panel BELOW this line.
 -- MERGE NOTE: Add declarations for Connect_Panel ABOVE this line.

begin -- Connect_Panel

 if Info.Panel_Id = Tae.Null_Panel_Id then
  Create_Panel
   ( Relative_Window     => Relative_Window,
     Panel_State         => Panel_State );
 else
  TAE.Tae_Wpt.Wpt_SetPanelState (Info.Panel_Id, Panel_State);
 end if;

 -- MERGE NOTE: Add code for Connect_Panel BELOW this line.
 -- MERGE NOTE: Add code for Connect_Panel ABOVE this line.

exception

 when TAE.Tae_Wpt.BAD_STATE =>
  Text_IO.Put_Line ("Panel_gui_in.Connect_Panel: "
   & "Invalid panel state.");
  raise;

end Connect_Panel;
```

```
-- ......................................................................
-- .
-- .  Destroy_Panel             -- Subprogram BODY
-- .
-- ......................................................................

procedure Destroy_Panel is

--| NOTES: (none)

 -- MERGE NOTE: Add declarations for Destroy_Panel BELOW this line.
 -- MERGE NOTE: Add declarations for Destroy_Panel ABOVE this line.

begin -- Destroy_Panel

 TAE.Tae_Wpt.Wpt_PanelErase(Info.Panel_Id);

 -- MERGE NOTE: Add code for Destroy_Panel BELOW this line.
 -- MERGE NOTE: Add code for Destroy_Panel ABOVE this line.

exception

 when TAE.Tae_Wpt.BAD_PANEL_ID =>
  Text_IO.Put_Line ("Panel_gui_in.Destroy_Panel: "
   & "Info.Panel_Id is an invalid id.");
  raise;

 when TAE.Tae_Wpt.ERASE_NULL_PANEL =>
  -- This panel has not been created yet, or has already been destroyed.
  -- Trap this exception and do nothing.
  null;

end Destroy_Panel;
```

```
--+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
--
-- begin EVENT HANDLERS
--
-- ....................................................................
-- .
-- .  choose_inputs_Event          -- Subprogram SPEC & BODY
-- .
-- ....................................................................

procedure choose_inputs_Event
 ( Info : in TAE.Tae_Wpt.Event_Context_Ptr ) is

--| PURPOSE:
--| EVENT HANDLER. Insert application specific information.
--|
--| NOTES: (none)

  Value : array (1..1) of String (1..TAE.Tae_Taeconf.STRINGSIZE);
  Count : TAE.Taeint;

  -- MERGE NOTE: Add declarations BELOW this line for parm: choose_inputs.
  -- MERGE NOTE: Add declarations ABOVE this line for parm: choose_inputs.

begin -- choose_inputs_Event

  TAE.Tae_Vm.Vm_Extract_Count (Info.Parm_Ptr, Count);
  if Count > 0 then
   TAE.Tae_Vm.Vm_Extract_SVAL (Info.Parm_Ptr, 1, Value(1));
  end if;

  -- MERGE NOTE: Add code BELOW this line for parm: choose_inputs.
--####
-- There are 4 possible input choices... Run, Pause, Add 5 Targets, or Quit
   -- Text_IO.put_line("....in pan_gui_in....");

  -- begin running or restarting the prototype
 if (tae_misc.s_equal(value(1), "Run")) then
   Text_IO.put_line("User selected 'Run'");
   choose_inputs_PKG.record_input(my_unit_PKG.Go);

  -- pause running the prototype
 elsif (tae_misc.s_equal(value(1), "Pause")) then
   Text_IO.put_line("User selected 'Pause'");
   choose_inputs_PKG.record_input(my_unit_PKG.Pause);

  -- add  targets to the target array
 elsif (tae_misc.s_equal(value(1), "Add Targets")) then
   Text_IO.put_line("User selected 'Add Targets'");
   choose_inputs_PKG.record_input(my_unit_PKG.Add_Target);


 elsif (tae_misc.s_equal(value(1), "Quit")) then
   Text_IO.put_line("User selected 'Quit', program shutting down...");
   global.Set_Application_Done; -- This will set a "done" flag to true
   choose_inputs_PKG.record_input(my_unit_PKG.Quit);

 else
   Text_IO.Put_line("Error in atacms.pan_gui_in_b.a, unknown selection");

 end if;


  -- MERGE NOTE: Add code ABOVE this line for parm: choose_inputs.

end choose_inputs_Event;
```

```
-- .......................................................................
-- .
-- .   header_Event                  -- Subprogram SPEC & BODY
-- .
-- .......................................................................

procedure header_Event
 ( Info : in TAE.Tae_Wpt.Event_Context_Ptr ) is

--| PURPOSE:
--| EVENT HANDLER. Insert application specific information.
--|
--| NOTES: (none)

 Value : array (1..1) of String (1..TAE.Tae_Taeconf.STRINGSIZE);
 Count : TAE.Taeint;

 -- MERGE NOTE: Add declarations BELOW this line for parm: header.
 -- MERGE NOTE: Add declarations ABOVE this line for parm: header.

begin -- header_Event

 TAE.Tae_Vm.Vm_Extract_Count (Info.Parm_Ptr, Count);
 if Count > 0 then
  TAE.Tae_Vm.Vm_Extract_SVAL (Info.Parm_Ptr, 1, Value(1));
 end if;

 -- MERGE NOTE: Add code BELOW this line for parm: header.
 -- MERGE NOTE: Add code ABOVE this line for parm: header.

end header_Event;
```

```
--
-- end EVENT HANDLERS
--
-------------------------------------------------------


-- ........................................................................
-- .
-- .  Dispatch_Item              -- Subprogram BODY
-- .
-- ........................................................................

procedure Dispatch_Item
 ( User_Context_Ptr : in TAE.Tae_Wpt.Event_Context_Ptr ) is

--| NOTES: (none)

begin -- Dispatch_Item

 if TAE.Tae_Misc.s_equal ("choose_inputs", User_Context_Ptr.Parm_Name) then
  choose_inputs_Event (User_Context_Ptr);
 elsif TAE.Tae_Misc.s_equal ("header", User_Context_Ptr.Parm_Name) then
  header_Event (User_Context_Ptr);
 end if;

end Dispatch_Item;


-- MERGE NOTE: Add additional code BELOW this line.
-- MERGE NOTE: Add additional code ABOVE this line.

end Panel_gui_in;
```

atacms.pan_gui_in_s.a

```
-- filename: atacms.pan_gui_in_s.a
--
-- Authors: Maj George Whitbeck and LCDR David Angrisani
-- Date:    6 Aug 96
-- Project: Thesis - A CAPS Prototype of the ATACMS C3 System
-- Purpose:

-- *** TAE Plus Code Generator version V5.3 [Merge Token: DO NOT DELETE.]
-- *** File:       pan_gui_in_s.a
-- *** Generated:  Jul 30 13:48:56 1996
-- ***************************************************************************
-- *
-- *   Panel_gui_in                -- Package SPEC
-- *
-- ***************************************************************************

with TAE;
with X_Windows;

package Panel_gui_in is

--| PURPOSE:
--| This package encapsulates the TAE Plus panel:  gui_in
--| These subprograms enable panel initialization, creation, destruction,
--| and event dispatching.  For more advanced manipulation of the panel
--| using the TAE package, the panel's Event_Context (Info) is provided.
--| It includes the Target and View (available after initialization)
--| and the Panel_Id (available after creation).
--|
--| INITIALIZATION EXCEPTIONS: (none)
--|
--| NOTES: (none)
--|
--| REGENERATED:
--| The following Workbench operations will cause regeneration of this file:
--|    The panel's name is changed (not title)
--| For panel:  gui_in
--|
--| CHANGE LOG:
--| 30-Jul-96   TAE    Generated


 Info : TAE.Tae_Wpt.Event_Context_Ptr;  -- panel information

 -- ......................................................................
 -- .
 -- .   Initialize_Panel            -- Subprogram SPEC
 -- .
 -- ......................................................................

 procedure Initialize_Panel
  ( Collection_Read             -- TAE Collection read from
    : in TAE.Tae_Co.Collection_Ptr );  -- resource file

 --| PURPOSE:
 --| This procedure initializes the Info.Target and Info.View for this panel
 --|
 --| EXCEPTIONS:
 --| TAE.UNINITIALIZED_PTR is raised if Collection_Read not initialized
 --| TAE.Tae_Co.NO_SUCH_MEMBER is raised if the panel is not in
 --|    Collection_Read
 --|
 --| NOTES: (none)
```

```
-- ......................................................................
-- .
-- .  Create_Panel                -- Subprogram SPEC
-- .
-- ......................................................................

procedure Create_Panel
  ( Panel_State                 -- Flags sent to Wpt_NewPanel.
     : in TAE.Tae_Wpt.Wpt_Flags
       := TAE.Tae_Wpt.WPT_PREFERRED;

    Relative_Window               -- Panel origin is offset from
     : in X_Windows.Window          -- this X Window.  Null_Window
       := X_Windows.Null_Window );     -- uses the root window.

--| PURPOSE:
--| This procedure creates this panel object in the specified Panel_State
--| and stores the panel Id in Info.Panel_Id.
--|
--| EXCEPTIONS:
--| TAE.UNINITIALIZED_PTR is raised if the panel is not initialized
--| TAE.TAE_FAIL is raised if the panel could not be created
--|
--| NOTES: (none)

-- ......................................................................
-- .
-- .  Connect_Panel               -- Subprogram SPEC
-- .
-- ......................................................................

procedure Connect_Panel
  ( Panel_State
     : in TAE.Tae_Wpt.Wpt_Flags
       := TAE.Tae_Wpt.WPT_PREFERRED;

    Relative_Window               -- Panel origin is offset from
     : in X_Windows.Window          -- this X Window.  Null_Window
       := X_Windows.Null_Window );     -- uses the root window.

--| PURPOSE:
--| If this panel doesn't exist, this procedure creates this panel object
--|  in the specifiec Panel_State and stores the panel Id in
--|  Info.Panel_Id.
--| If this panel does exist, it is set to the specified Panel_State.
--|  In this case, Relative_Window is ignored.
--|
--| EXCEPTIONS:
--| TAE.UNINITIALIZED_PTR is raised from Create_Panel if the panel is
--|   not initialized
--| TAE.TAE_FAIL is raised from Create_Panel if the panel could not be
--|   created
--| TAE.Tae_Wpt.BAD_STATE is raised if the panel exists and the
--|   Panel_State is an invalid state
--|
--| NOTES: (none)

-- ......................................................................
-- .
-- .  Destroy_Panel               -- Subprogram SPEC
-- .
-- ......................................................................

procedure Destroy_Panel;

--| PURPOSE:
--| This procedure erases a panel from the screen and de-allocates the
--| associated panel object (not the target and view).
--|
--| EXCEPTIONS:
--| TAE.Tae_Wpt.BAD_PANEL_ID is raised if Info.Panel_Id is an invalid id.
```

```
--|
--| NOTES:
--| Info.Panel_Id is set to TAE.NULL_PANEL_ID, and should not referenced
--| in any Wpt call until it is created again.


-- ......................................................................
-- .
-- .   Dispatch_Item               -- Subprogram SPEC
-- .
-- ......................................................................

procedure Dispatch_Item
  ( User_Context_Ptr                -- Wpt Event Context for a PARM
    : in TAE.Tae_Wpt.Event_Context_Ptr ); -- event.

--| PURPOSE:
--| This procedure calls the Event Handler specified by User_Context_Ptr
--|
--| EXCEPTIONS:
--| Application-specific
--|
--| NOTES: (none)

end Panel_gui_in;
```

atacms.scdl_link_op.a

```
-- filename: atacms.scdl_link_op.a
--
-- Authors:  Maj George Whitbeck and LCDR David Angrisani
-- Date:     6 Aug 96;modified 04 Sep 96
-- Project:  Thesis - A CAPS Prototype of the ATACMS C3 System
-- Purpose:  This comm link will pass on 90% of traffic and corrupt 10% due to
--           simulated friction
-- to compile  type... ada atacms.scdl_link_op.a

with text_io; use text_io;
with global_random_PKG;
with jstars_array_PKG;
with constants_PKG;

package scdl_link_op_PKG is

  procedure scdl_link_op
     (target_array1_str : in jstars_array_PKG.jstars_array;
      target_array2_str : out jstars_array_PKG.jstars_array );

end scdl_link_op_PKG;


package body scdl_link_op_PKG is

 procedure scdl_link_processing_delay is
 begin
   delay duration(constants_PKG.mean_scdl_link_processing_delay);--temporary;use mean & deviation to
                                                    --calculate delay(see constants_PKG)
 end scdl_link_processing_delay;

 procedure scdl_link_transmission_prep_delay is
 begin
   delay duration(constants_PKG.mean_scdl_link_transmission_prep_delay);--temporary;use mean &
                                                                              --deviation to calculate
                                                                              --delay(see constants_PKG)
 end scdl_link_transmission_prep_delay;


 procedure scdl_link_op
   (target_array1_str : in jstars_array_PKG.jstars_array;
    target_array2_str : out jstars_array_PKG.jstars_array) is

         package flt_io is new float_io(float);
    use flt_io;
         my_random : float := 0.0;

 begin
         --put_line("Now in procedure 'scdl_link_op'");
    --put("Good xmission so far? ... ");
    --constants_PKG.bool_io.put(target_array1_str(constants_PKG.min_array,
    --                 constants_PKG.min_array).good_xmission); new_line;


         -- a direct copy of the array of records
         target_array2_str := target_array1_str;

         -- below is simulated error; error types not defined
         my_random := global_random_PKG.ftn_global_random;

         if my_random > constants_PKG.crash_rate then  -- set flag

           target_array2_str(constants_PKG.min_array,constants_PKG.min_array).good_xmission:= false;
           --put_line("Still in procedure 'scdl_link_op' ... crash occured");

         end if;
```

```
--THIS SIMULATES THE LATENCY FROM JSTARS TO THE GROUND STATION MODULE
            delay 3.0;

--THESE ARE PLACEHOLDERS; USE/MODIFY AS NEEDED
      --scdl_link_processing_delay;
      --scdl_link_transmission_prep_delay;

  end scdl_link_op;

end scdl_link_op_PKG;
```

atacms.shooter_op.a

```
-- filename: atacms.shooter_op.a
--
-- Authors:  Maj George Whitbeck and LCDR David Angrisani
-- Date:     6 Aug 96; modified 04 Sep 96
-- Project:  Thesis - A CAPS Prototype of the ATACMS C3 System
-- Purpose:  This package simulates the operations of the ground
--           Command Station
--
-- to compile... ada atacms.shooter_op.a



with text_io; use text_io;
with target_data_PKG;
with constants_PKG;

package shooter_op_PKG is

 procedure shooter_op (fire_cmd4_str : target_data_PKG.target_data;
               gui_out_str  : out target_data_PKG.target_data);

end shooter_op_PKG;



package body shooter_op_PKG is


 procedure shooter_operator_delay is
 begin
   delay duration(constants_PKG.mean_shooter_operator_delay); --temporary;use mean & deviation
                                                              --to calculate delay
                                                              --(see constants_PKG)
 end shooter_operator_delay;

 procedure shooter_processing_delay is
 begin
   delay duration(constants_PKG.mean_shooter_processing_delay);--temporary;use mean & deviation
                                                              --to calculate delay(see
                                                              --constants_PKG)
 end shooter_processing_delay;

 procedure shooter_transmission_prep_delay is
 begin
   delay duration(constants_PKG.mean_shooter_transmission_prep_delay);--temporary;use mean &
                                                                     --deviation to calculate
                                                                     --delay(see constants_PKG)
 end shooter_transmission_prep_delay;

 procedure shooter_error is
 begin
   null;                              --insert simulated errors
 end shooter_error;




procedure shooter_op (fire_cmd4_str : target_data_PKG.target_data;
              gui_out_str  : out target_data_PKG.target_data) is
```

```
begin

 put_line("Transmission received by shooter, sending data to display panel.  ");

         if fire_cmd4_str.tgt_num = 0 then
                   put_line("Just received a communications check only.  No mission assigned. ");

         elsif fire_cmd4_str.good_xmission then
                   put_line("Just received a Good Transmission. ");
                   put("Mission successfully being fired at Target Number ");
                   constants_PKG.int_io.put(fire_cmd4_str.tgt_num,0); new_line;

         else
                   put("Detected a crash. Mission lost.  Target number ");
                   constants_PKG.int_io.put(fire_cmd4_str.tgt_num,0);
                   put(" was lost.");
         end if;

 new_line(3);

 gui_out_str := fire_cmd4_str;


--THESE ARE PLACEHOLDERS; USE/MODIFY AS NECESSARY
  --shooter_operator_delay;
  --shooter_processing_delay;
  --shooter_transmission_prep_delay;
  --shooter_error;


 end shooter_op;


end shooter_op_PKG;


-- target_data_PKG.int_io.put(fire_cmd4_str.easting,0);
```

atacms.target_data.a

```
-- filename: atacms.target_data.a
--
-- Authors:  Maj George Whitbeck and LCDR David Angrisani
-- Date:    6 Aug 96
-- Project:  Thesis - A CAPS Prototype of the ATACMS C3 System
-- Purpose:


-- This file is where I'd like to declare my target data record
-- that I will use as a stream in my thesis
-- I avoided the use clause as much as possible.  Also, note naming
-- convention here with filename, package name, and type name!!!!!

-- to just compile type... ada atacms.target_data.a


with text_io;
use text_io;


package target_data_PKG is

  -- to instantiate integer generic
  package int_io is new integer_io (integer);

  -- create enumerated type for artillery method of control
  -- and allow i/o
  type method_cntl_type is (WR, AMC, DNL, TOT);
  package cntl_io is new text_io.enumeration_io(method_cntl_type);

  -- to create legal entries for the UTM coordinate system
  subtype east_coord is integer range 0..999_999;
  subtype north_coord is integer range 0..9_999_999;
  subtype alt_coord is integer range -1000..10_000;

type target_data is record
      target_class: integer:=0;
      easting : east_coord := 0;
      northing : north_coord := 0;
      alt : alt_coord := 0;
      tgt_num : natural := 0;
      tgt_desc : string(1..60) := (others => '*');
      method_cntl : method_cntl_type := DNL;
      good_xmission : boolean := true;
      status : natural := 0;

end record;

end target_data_PKG;
```

atacms.target_emitter_array.a

```
-- filename: atacms.package target_emitter_array.a
--
-- Authors:  Maj George Whitbeck and LCDR David Angrisani
-- Date:     6 Aug 96
-- Project:  Thesis - A CAPS Prototype of the ATACMS C3 System
-- Purpose:  This file is where we declare an array  to simulate radar
--           relections that we will use as a stream in our thesis
--

-- to just compile type... ada atacms.target_emitter_array.a

with text_io;
use text_io;
with constants_PKG;

package target_emitter_array_PKG is


 --THIS SECTION CONTAINS target_emitter_array, which is the type for the
 -- output stream for target_emitter_op and the
 -- input stream for jstars_op

 -- to create legal entries for the UTM coordinate system
  subtype east_coord is integer range 0..999_999;
  subtype north_coord is integer range 0..9_999_999;
  subtype alt_coord is integer range -1000..10_000;

 type target_emitter_array_record is record
  target_class: integer:=0;
  easting : east_coord := 0;
  northing : north_coord := 0;
  tgt_num : natural := 0;
  good_xmission : boolean := true;
  status : natural := 0;

 end record;

 type target_emitter_array is array(constants_PKG.min_array..constants_PKG.max_array,
                       constants_PKG.min_array..constants_PKG.max_array)
                       of target_emitter_array_record;


end target_emitter_array_PKG;
```

atacms.target_emitter_op.a

```
-- filename:  atacms.target_emitter_op.a
--
-- Authors:  Maj George Whitbeck and LCDR David Angrisani
-- Date:    6 Aug 96
-- Project:  Thesis - A CAPS Prototype of the ATACMS C3 System
-- Purpose:  This package generates the target emmissions which will be
--          detected by the JSTARS platform.  This assigns information
--          which could be discerned from the sensor itself; in
--          this case, a target class is assigned.
--
-- to just compile... ada atacms.target_emitter_op.a


with text_io; use text_io;
with target_emitter_array_PKG;   -- to instantiate our array of record type
with global_random_PKG;
with constants_PKG;
with my_unit_PKG;

package target_emitter_op_PKG is

 procedure target_emitter_op
      (gui_in_str : my_unit_PKG.my_unit;
       emission_str  : out target_emitter_array_PKG.target_emitter_array);

end target_emitter_op_PKG;


package body target_emitter_op_PKG is

 -- create a sample emitter array to manipulate
 my_target_emitter_array: target_emitter_array_PKG.target_emitter_array;


 procedure target_emitter_op
   (gui_in_str : my_unit_PKG.my_unit;
    emission_str  : out target_emitter_array_PKG.target_emitter_array) is


   rand_num: natural:=0;  -- to determine to target class
   lv_int : natural;
   row_x : natural := 0;
   column_y : natural := 0;

 begin



   --put("Now in procedure 'target_emitter`,");

  lv_int := my_unit_PKG.my_unit'pos(gui_in_str);
   --constants_PKG.int_io.put(lv_int,0);
   new_line;

  if lv_int = 2 then

   Put_line("Reflections from multiple targets detected.");
   for x in 1..constants_PKG.max_new_tgts loop

     rand_num := integer(100.0*global_random_PKG.ftn_global_random);
     row_x := constants_PKG.min_array + (integer (10.0*global_random_PKG.ftn_global_random));
     column_y := constants_PKG.min_array + rand_num/10;  -- avoid 3d ftn call, reuse rand_num
```

```
case rand_num is
            when 0..50 =>
                    my_target_emitter_array(row_x,column_y).target_class:= 1;
            when 51..60 =>
                    my_target_emitter_array(row_x,column_y).target_class:= 2;
            when 61..70 =>
                    my_target_emitter_array(row_x,column_y).target_class:= 3;
            when 71..80 =>
                    my_target_emitter_array(row_x,column_y).target_class:= 4;
    when 81..99 =>
                    my_target_emitter_array(row_x,column_y).target_class:= 5;
            when others =>
                    null;
    end case;

  end loop;
 end if;

emission_str:= my_target_emitter_array;  -- copy local array to output parameter

end target_emitter_op;

end target_emitter_op_PKG;

  -- put("Now in procedure 'target_emitter', just added a target class of ");
  -- constants_PKG.int_io.put(my_target_emitter_array(row_x,column_y).target_class,0);
  -- new_line;
```

APPENDIX C. REFINEMENT II GRAPH AND PSDL

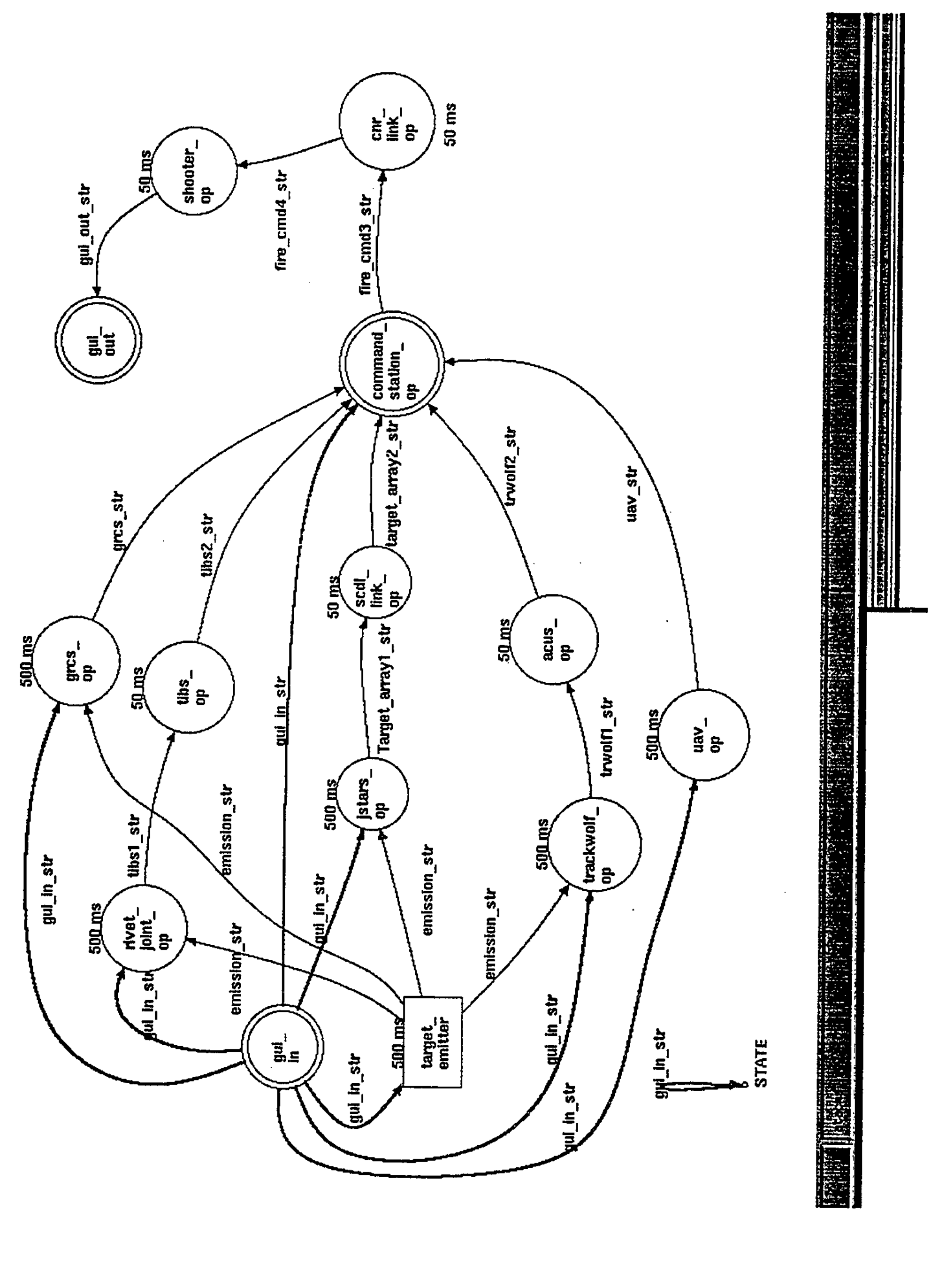


Figure 1. Refinement II Model - Top Level.

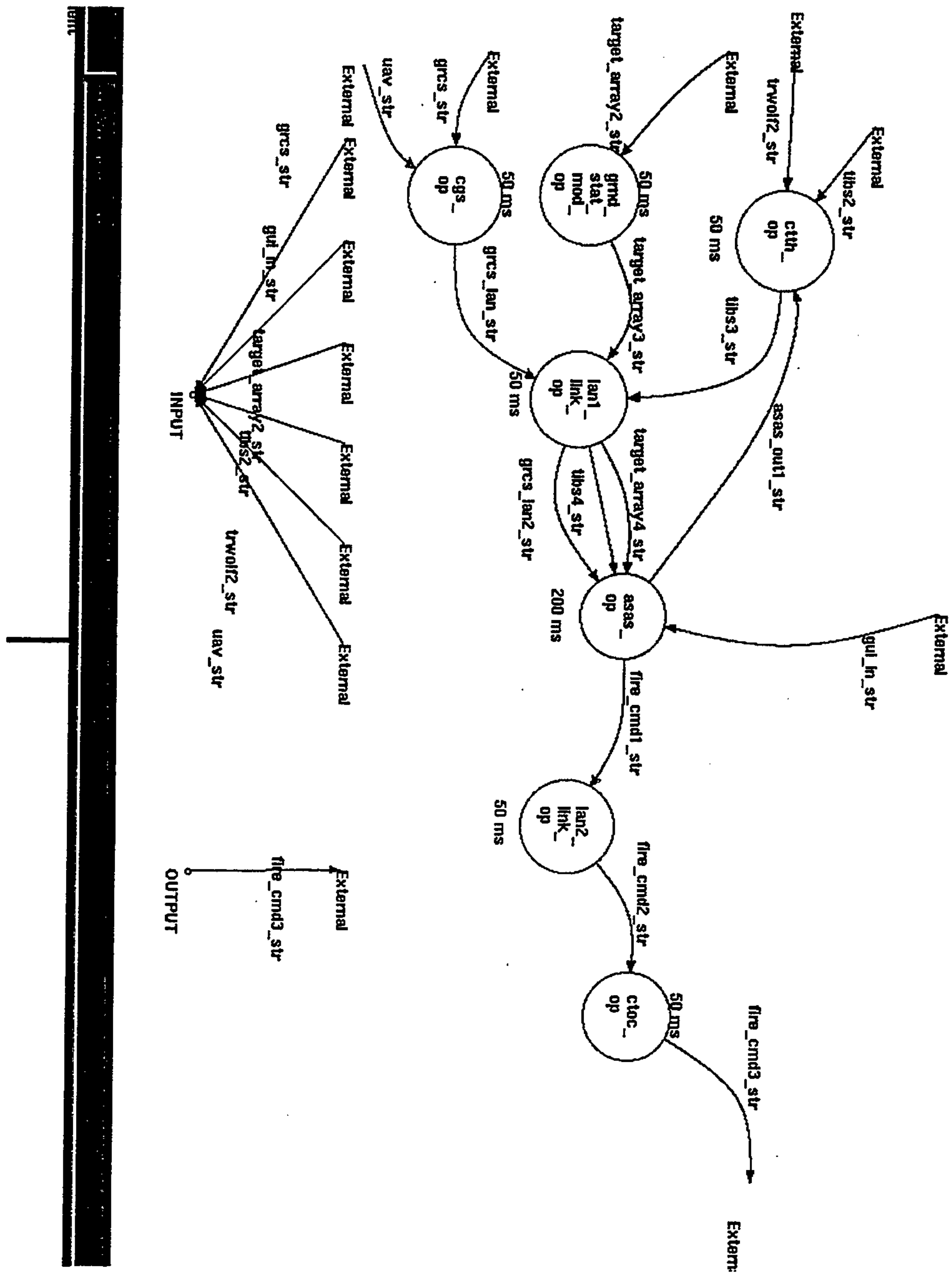


Figure 2. Refinement II Model - Decomposed Command Station.

```
TYPE grnd_stat_mod_array
 SPECIFICATION
END
IMPLEMENTATION ADA grnd_stat_mod_array

END

TYPE jstars_array
 SPECIFICATION
END
IMPLEMENTATION ADA jstars_array

END

TYPE my_unit
 SPECIFICATION
  OPERATOR pause
   SPECIFICATION
    OUTPUT
     x : my_unit
  END
END
IMPLEMENTATION ADA my_unit

END

TYPE target_data
 SPECIFICATION
END
IMPLEMENTATION ADA target_data

END

TYPE target_emitter_array
 SPECIFICATION
END
IMPLEMENTATION ADA target_emitter_array

END

OPERATOR acus_op
 SPECIFICATION
  INPUT
   trwolf1_str : UNDEFINED_TYPE_NAME
  OUTPUT
   trwolf2_str : UNDEFINED_TYPE_NAME
  MAXIMUM EXECUTION TIME 50 MS
END

OPERATOR asas_op
 SPECIFICATION
  INPUT
   grcs_lan2_str : UNDEFINED_TYPE_NAME,
   gui_in_str : my_unit,
   target_array4_str : grnd_stat_mod_array,
   tibs4_str : UNDEFINED_TYPE_NAME
  OUTPUT
   asas_out1_str : UNDEFINED_TYPE_NAME,
   fire_cmd1_str : target_data
  MAXIMUM EXECUTION TIME 200 MS
END
IMPLEMENTATION ADA asas_op

END

OPERATOR atacms
 SPECIFICATION
  STATES
   gui_in_str : my_unit
   INITIALLY
    pause
END
IMPLEMENTATION
 GRAPH
  VERTEX acus_op : 50 MS

  VERTEX cnr_link_op : 50 MS

  VERTEX command_station_op

  VERTEX grcs_op : 500 MS

  VERTEX gui_in

  VERTEX gui_out

  VERTEX jstars_op : 500 MS

  VERTEX rivet_joint_op : 500 MS

  VERTEX scdl_link_op : 50 MS

  VERTEX shooter_op : 50 MS

  VERTEX target_emitter : 500 MS

  VERTEX tibs_op : 50 MS

  VERTEX trackwolf_op : 500 MS

  VERTEX uav_op : 500 MS

  EDGE Target_array1_str
   jstars_op ->
   scdl_link_op

  EDGE emission_str
   target_emitter ->
   trackwolf_op

  EDGE emission_str
   target_emitter ->
   grcs_op

  EDGE emission_str
   target_emitter ->
   rivet_joint_op

  EDGE emission_str
   target_emitter ->
   jstars_op

  EDGE fire_cmd3_str
   command_station_op ->
   cnr_link_op

  EDGE fire_cmd4_str
   cnr_link_op ->
   shooter_op

  EDGE grcs_str
   grcs_op ->
   command_station_op
```

```
EDGE gui_in_str
 gui_in ->
 uav_op

EDGE gui_in_str
 gui_in ->
 rivet_joint_op

EDGE gui_in_str
 gui_in ->
 grcs_op

EDGE gui_in_str
 gui_in ->
 trackwolf_op

EDGE gui_in_str
 gui_in ->
 target_emitter

EDGE gui_in_str
 gui_in ->
 jstars_op

EDGE gui_in_str
 gui_in ->
 command_station_op

EDGE gui_out_str
 shooter_op ->
 gui_out

EDGE target_array2_str
 scdl_link_op ->
 command_station_op

EDGE tibs1_str
 rivet_joint_op ->
 tibs_op

EDGE tibs2_str
 tibs_op ->
 command_station_op

EDGE trwolf1_str
 trackwolf_op ->
 acus_op

EDGE trwolf2_str
 acus_op ->
 command_station_op

EDGE uav_str
 uav_op ->
 command_station_op
DATA STREAM
 Target_array1_str : UNDEFINED_TYPE_NAME,
 emission_str : UNDEFINED_TYPE_NAME,
 fire_cmd3_str : target_data,
 fire_cmd4_str : target_data,
 grcs_str : UNDEFINED_TYPE_NAME,
 gui_out_str : target_data,
 target_array2_str : jstars_array,
 tibs1_str : UNDEFINED_TYPE_NAME,
 tibs2_str : UNDEFINED_TYPE_NAME,
 trwolf1_str : UNDEFINED_TYPE_NAME,
 trwolf2_str : UNDEFINED_TYPE_NAME,
 uav_str : UNDEFINED_TYPE_NAME

 CONTROL CONSTRAINTS
  OPERATOR acus_op

  OPERATOR cnr_link_op
   TRIGGERED BY SOME
    fire_cmd3_str

  OPERATOR command_station_op

  OPERATOR grcs_op

  OPERATOR gui_in

  OPERATOR gui_out

  OPERATOR jstars_op
   TRIGGERED IF
    gui_in_str /= my_unit.pause
   PERIOD 8000 MS

  OPERATOR rivet_joint_op

  OPERATOR scdl_link_op
   TRIGGERED BY SOME
    target_array1_str

  OPERATOR shooter_op
   TRIGGERED BY SOME
    fire_cmd4_str

  OPERATOR target_emitter

  OPERATOR tibs_op

  OPERATOR trackwolf_op

  OPERATOR uav_op
END

OPERATOR cgs_op
 SPECIFICATION
  INPUT
   grcs_str : UNDEFINED_TYPE_NAME,
   uav_str : UNDEFINED_TYPE_NAME
  OUTPUT
   grcs_lan_str : UNDEFINED_TYPE_NAME
  MAXIMUM EXECUTION TIME 50 MS
END

OPERATOR choose_inputs
 SPECIFICATION
  OUTPUT
   gui_in_str : my_unit
  MAXIMUM EXECUTION TIME 200 MS
END
IMPLEMENTATION ADA choose_inputs

END

OPERATOR cmds_out
 SPECIFICATION
  INPUT
   gui_out_str : target_data
END
IMPLEMENTATION ADA cmds_out

END
```

```
OPERATOR cnr_link_op
 SPECIFICATION
  INPUT
   fire_cmd3_str : target_data
  OUTPUT
   fire_cmd4_str : target_data
  MAXIMUM EXECUTION TIME 50 MS
END
IMPLEMENTATION ADA cnr_link_op

END

OPERATOR command_station_op
 SPECIFICATION
  INPUT
   grcs_str : UNDEFINED_TYPE_NAME,
   gui_in_str : my_unit,
   target_array2_str : jstars_array,
   tibs2_str : UNDEFINED_TYPE_NAME,
   trwolf2_str : UNDEFINED_TYPE_NAME,
   uav_str : UNDEFINED_TYPE_NAME
  OUTPUT
   fire_cmd3_str : target_data
END
IMPLEMENTATION
 GRAPH
  VERTEX asas_op : 200 MS

  VERTEX cgs_op : 50 MS

  VERTEX ctoc_op : 50 MS

  VERTEX ctth_op : 50 MS

  VERTEX grnd_stat_mod_op : 50 MS

  VERTEX lan1_link_op : 50 MS

  VERTEX lan2_link_op : 50 MS

  EDGE asas_out1_str
   asas_op ->
   ctth_op

  EDGE fire_cmd1_str
   asas_op ->
   lan2_link_op

  EDGE fire_cmd2_str
   lan2_link_op ->
   ctoc_op

  EDGE fire_cmd3_str
   ctoc_op ->
   EXTERNAL

  EDGE grcs_lan2_str
   lan1_link_op ->
   asas_op

  EDGE grcs_lan_str
   cgs_op ->
   lan1_link_op

  EDGE grcs_str
   EXTERNAL ->
   cgs_op

  EDGE gui_in_str
   EXTERNAL ->
   asas_op

  EDGE target_array2_str
   EXTERNAL ->
   grnd_stat_mod_op

  EDGE target_array3_str
   grnd_stat_mod_op ->
   lan1_link_op

  EDGE target_array4_str
   lan1_link_op ->
   asas_op

  EDGE tibs2_str
   EXTERNAL ->
   ctth_op

  EDGE tibs3_str
   ctth_op ->
   lan1_link_op

  EDGE tibs4_str
   lan1_link_op ->
   asas_op

  EDGE trwolf2_str
   EXTERNAL ->
   ctth_op

  EDGE uav_str
   EXTERNAL ->
   cgs_op
 DATA STREAM
  asas_out1_str : UNDEFINED_TYPE_NAME,
  fire_cmd1_str : target_data,
  fire_cmd2_str : target_data,
  grcs_lan2_str : UNDEFINED_TYPE_NAME,
  grcs_lan_str : UNDEFINED_TYPE_NAME,
  target_array3_str : grnd_stat_mod_array,
  target_array4_str : grnd_stat_mod_array,
  tibs3_str : UNDEFINED_TYPE_NAME,
  tibs4_str : UNDEFINED_TYPE_NAME
 CONTROL CONSTRAINTS
  OPERATOR asas_op
   TRIGGERED IF
    gui_in_str /= my_unit.pause
   PERIOD 4000 MS

  OPERATOR cgs_op

  OPERATOR ctoc_op
   TRIGGERED BY SOME
    fire_cmd2_str

  OPERATOR ctth_op

  OPERATOR grnd_stat_mod_op
   TRIGGERED BY SOME
    target_array2_str
   MAXIMUM RESPONSE TIME 5050 MS

  OPERATOR lan1_link_op
   TRIGGERED BY SOME
    target_array3_str
```

```
   OPERATOR lan2_link_op
    TRIGGERED BY SOME
     fire_cmd1_str
 END

 OPERATOR ctoc_op
  SPECIFICATION
   INPUT
    fire_cmd2_str : target_data
   OUTPUT
    fire_cmd3_str : target_data
   MAXIMUM EXECUTION TIME 50 MS
 END
 IMPLEMENTATION ADA ctoc_op

 END

 OPERATOR ctth_op
  SPECIFICATION
   INPUT
    asas_out1_str : UNDEFINED_TYPE_NAME,
    tibs2_str : UNDEFINED_TYPE_NAME,
    trwolf2_str : UNDEFINED_TYPE_NAME
   OUTPUT
    tibs3_str : UNDEFINED_TYPE_NAME
   MAXIMUM EXECUTION TIME 50 MS
 END

 OPERATOR grcs_op
  SPECIFICATION
   INPUT
    emission_str : UNDEFINED_TYPE_NAME,
    gui_in_str : my_unit
   OUTPUT
    grcs_str : UNDEFINED_TYPE_NAME
   MAXIMUM EXECUTION TIME 500 MS
 END

 OPERATOR grnd_stat_mod_op
  SPECIFICATION
   INPUT
    target_array2_str : jstars_array
   OUTPUT
    target_array3_str : grnd_stat_mod_array
   MAXIMUM EXECUTION TIME 50 MS
 END
 IMPLEMENTATION ADA grnd_stat_mod_op

 END

 OPERATOR gui_in
  SPECIFICATION
   OUTPUT
    gui_in_str : my_unit
 END
 IMPLEMENTATION
  GRAPH
   VERTEX choose_inputs : 200 MS

   VERTEX gui_input_event_monitor : 200 MS

   EDGE gui_in_str
    choose_inputs ->
    EXTERNAL
  CONTROL CONSTRAINTS
   OPERATOR choose_inputs
    PERIOD 2000 MS

   OPERATOR gui_input_event_monitor
 END

 OPERATOR gui_input_event_monitor
  SPECIFICATION
   MAXIMUM EXECUTION TIME 200 MS
 END
 IMPLEMENTATION ADA gui_input_event_monitor

 END

 OPERATOR gui_out
  SPECIFICATION
   INPUT
    gui_out_str : target_data
 END
 IMPLEMENTATION
  GRAPH
   VERTEX cmds_out

   EDGE gui_out_str
    EXTERNAL ->
    cmds_out
  CONTROL CONSTRAINTS
   OPERATOR cmds_out
    TRIGGERED BY SOME
     gui_out_str
 END

 OPERATOR jstars_op
  SPECIFICATION
   INPUT
    emission_str : UNDEFINED_TYPE_NAME,
    gui_in_str : my_unit
   OUTPUT
    Target_array1_str : UNDEFINED_TYPE_NAME
   MAXIMUM EXECUTION TIME 500 MS
 END
 IMPLEMENTATION ADA jstars_op

 END

 OPERATOR lan1_link_op
  SPECIFICATION
   INPUT
    grcs_lan_str : UNDEFINED_TYPE_NAME,
    target_array3_str : grnd_stat_mod_array,
    tibs3_str : UNDEFINED_TYPE_NAME
   OUTPUT
    grcs_lan2_str : UNDEFINED_TYPE_NAME,
    target_array4_str : grnd_stat_mod_array,
    tibs4_str : UNDEFINED_TYPE_NAME
   MAXIMUM EXECUTION TIME 50 MS
 END
 IMPLEMENTATION ADA lan1_link_op

 END

 OPERATOR lan2_link_op
  SPECIFICATION
   INPUT
    fire_cmd1_str : target_data
   OUTPUT
    fire_cmd2_str : target_data
   MAXIMUM EXECUTION TIME 50 MS
 END
 IMPLEMENTATION ADA lan2_link_op

 END
```

```
OPERATOR rivet_joint_op
 SPECIFICATION
  INPUT
   emission_str : UNDEFINED_TYPE_NAME,
   gui_in_str : my_unit
  OUTPUT
   tibs1_str : UNDEFINED_TYPE_NAME
  MAXIMUM EXECUTION TIME 500 MS
END

OPERATOR scdl_link_op
 SPECIFICATION
  INPUT
   Target_array1_str : UNDEFINED_TYPE_NAME
  OUTPUT
   target_array2_str : jstars_array
  MAXIMUM EXECUTION TIME 50 MS
END
IMPLEMENTATION ADA scdl_link_op

END

OPERATOR shooter_op
 SPECIFICATION
  INPUT
   fire_cmd4_str : target_data
  OUTPUT
   gui_out_str : target_data
  MAXIMUM EXECUTION TIME 50 MS
END
IMPLEMENTATION ADA shooter_op

END

OPERATOR target_emitter
 SPECIFICATION
  INPUT
   gui_in_str : my_unit
  OUTPUT
   emission_str : UNDEFINED_TYPE_NAME
  MAXIMUM EXECUTION TIME 500 MS
END

OPERATOR target_emitter_op
 SPECIFICATION
  MAXIMUM EXECUTION TIME 500 MS
END
IMPLEMENTATION ADA target_emitter_op

END

OPERATOR tibs_op
 SPECIFICATION
  INPUT
   tibs1_str : UNDEFINED_TYPE_NAME
  OUTPUT
   tibs2_str : UNDEFINED_TYPE_NAME
  MAXIMUM EXECUTION TIME 50 MS
END

OPERATOR trackwolf_op
 SPECIFICATION
  INPUT
   emission_str : UNDEFINED_TYPE_NAME,
   gui_in_str : my_unit
  OUTPUT
   trwolf1_str : UNDEFINED_TYPE_NAME
  MAXIMUM EXECUTION TIME 500 MS
END

OPERATOR uav_op
 SPECIFICATION
  INPUT
   gui_in_str : my_unit
  OUTPUT
   uav_str : UNDEFINED_TYPE_NAME
  MAXIMUM EXECUTION TIME 500 MS
END
```

APPENDIX D. CAPS MINI-TUTORIALS

The following are a series of mini-tutorials which describe various CAPS actions, procedures, and concepts. These will serve as a basis for beginning the CAPS documentation process. An index of subjects is provided. All examples are based on the prototype. Reference the graphs and PSDL files in Appendix B to see where these operations take place.

1. Changing or Adding a Type in a Working Prototype
2. Creating Functionality
3. Removing a Bubble from a Working Prototype.
4. Decomposing a Working Bubble
5. Adding a Quit Button
6. Making a new version of a prototype

1. Changing or Adding a Type in a Working Prototype

This example is changing from a system defined type (like Integer) to a user defined type (like Target_data). We changed the tgt_num_steam in TACMS1 from an integer to a user defined type which we created and tested

In the PSDL editor do this:

1. Tell CAPS you'll be using a user-defined type. Invoke the PSDL editor, go to the blank line at the top, and click an empty space. Note the entry "psdl_components" appear at bottom, click it. Note the entry "type" appear at bottom, click it, see the TYPE structure at the top. Type in the word "target_data" (or the name of your new type) at the top, press Enter. Three lines below, click on <type implementation>, then at the bottom click on "Ada implementation". Click off to the right of "Ada Implementation" to propagate the change. The task is done unless you need extra fancy stuff like adding the declaration of a constant for defining an initial state for state machines.

2. Now, change a stream to the new type (similar steps for first time declaration): Go to the DATA STREAMS declaration in the root node section in the PSDL, find the stream in question, and click on the old type - it will now be underlined. Edit/cut structure, then either <decl_type_name> or <identifier> appears in its place.

a. If you cut a valid type, <decl_type_name> appears in its place. At bottom, click on "user_defined" and <decl_type_name> above changes to <identifier>. Above now type in the word "target_data" (or your type name) and hit Return. Now click anywhere in the open space of the window to the right to propagate the change.

b. If, however, you cut UNDEFINED_TYPE_NAME, then <identifier> appears in its place. Just type in the new type name and press return.

check your work...
save and exit
done with PSDL changes

3. If you are using a user defined type, you'll have to create a file which defines the type. So, create a file called "<prototype>.<type_name>.a". This file will contain your record or abstract data type. It will contain a package called "<type_name>_PKG". Inside the package, you will create the type starting with a statement like "type <type_name> is record" ... "end record". If your type has any operations, their implementations also go in this package.

2. Creating Functionality

This section shows you how to make the stub files for your CAPS projects. Note the EXACT NAMING CONVENTIONS. Fill in the names for your prototype, operator, stream, parameter mode and stream_type below. The filename must also be correct, Ada is not case sensitive but UNIX is! Name the file: <prototype>.<operator_name>.a i.e. "weather.temperature.a". What follows is a skeleton ada file.

```
-- Be sure to "with" any needed packages, SPECIFICALLY any user defined
  types!!!
with my_user_defined_type_PKG; -- an example of a user defined type
with text_io;                  -- only use as needed


package <operator_name>_PKG is
  procedure <operator_name> (<stream_name> : in or out <stream_type> );
end <operator_name>_PKG;

package body <operator_name>_PKG is
 procedure <operator_name> (<stream_name> : in or out <stream_type>  ) is
 begin
       null;           -- YOU put functionality here
 end <operator_name>;
end <operator_name>_PKG;
```

3. Removing a Bubble from a Working Prototype.

Start in the graphics editor. Select the bubble you want to kill then press delete. Now return to the SDE and look for remnants of the old operator. Expect to see error messages about multiple roots. Find the old bubble and click on the adjacent word OPERATOR and all associated parts will be underlined. Select Edit/Cut_structure and the underlined parts will be gone. Click in the open area to the right in the main window to propagate the change.

If the bubble was a child, inspect the PSDL of its parent to ensure all trash is gone. It should be. Also look for references to Operator NONAME_##. This is a sure sign that you've confused the SDE.

Save and exit. Next, edit the interface. Remove components that were tied to the bubble you deleted. Also, check the ADA code for any components tied to the bubble deleted because their interfaces may have changed due to deleted streams. Regenerate code per normal steps.

4. Decomposing a Working Bubble

If you have a working prototype and you want to decompose one of the bubbles, here is a checklist of all the things you must do. Note: click is a single press and release of the left mouse button. Click and drag is a single press without letting go until you've moved to where you want to go.

- Open the PSDL editor for your prototype.
- Select edit-graph.
- Place your mouse pointer on the rim of the bubble you want to decompose then click.
- Select graph/decompose.
- You'll get a new window. Note the streams from your previous window are now at the bottom. You must assign each of these to your new children bubbles. Those labeled "INPUT" must start outside a bubble and end in one. Those labeled "OUTPUT" are just the opposite.
- Create operators per normal rules.
- For an external to input stream, select (click) the stream icon, move your cursor to a point outside a bubble, click, then move your cursor to a point inside a bubble, click again. Label the new stream per normal rules but use the name of the stream from the bottom of the screen.
- For an input stream to external, select (click) the stream icon, move your cursor to a point inside a bubble, click, then move your cursor to a point outside a bubble, DOUBLE click. Label the new stream per normal rules but use the name of the stream from the bottom of the screen.

Connect the above bubbles per normal rules. Don't forget to give names to these connecting streams. Give your new child bubbles names and MET's (if needed). Of course, if you're prototyping a real system, give real names. When done, select graph/save_and_continue then graph/edit_parent. The screen will return to your prototype screen. The bubble you decomposed will now have a double circle. Nothing to do here so just select graph/return_to_SDE. INSPECT YOUR PSDL specifically look for the string "NONAME_##" where ## is some integer. This means you forgot to label something so CAPS did it for you. Go back to editing the graph, select the offender, and change its name from NONAME_## to whatever you intended. If you don't want it, select it then press delete. When you again return to PSDL, the problem should be fixed.

NOW, begin editing your PSDL as follows:

Go to your parent bubble in the PSDL. Note that it is much bigger and contains information about it's children and the streams that connect them. First, assign a type to the internal, connecting streams. Click on "<decl_type_name>". At the bottom, click on the appropriate type, press return, then click to the right of the newly entered information to propagate the change.

While still working in the area of the parent, assign control constraints to the children as needed. Use normal rules. DON'T FORGET to trigger operators as needed. Now move to one of the new children in the PSDL. All types should be filled in since you just did that in the parent area and the changes should have propagated. But, note that the last line that reads <operator implementation>. Click on that. Note the two choices that appear at the bottom. Click on "*ADA_ Implementation.*" In the PSDL, "<operator implementation>" changes to "IMPLEMENTATION ADA <operator name>". [As a general rule, children bubbles have an "ADA_ Implementation" which means that you will write an Ada package for them using the following naming convention:"<prototype_name>. <operator_name>.a". This only applies to the children bubbles you create as a result of decomposing a parent. Don't confuse this with the fact that your overall prototype is a parent and all bubbles are actually its children.] INSPECT THE PSDL AGAIN. You should have only one root operator and nothing in braces "< >".

Save and exit.

Go to your files for this prototype.

You may already have a ".a" file for the bubble you just decomposed. You don't need it anymore but you may need to transfer it's functionality to some of it's children. They all need ".a" files and you must write them from scratch. Section C of this chapter covers this.

5. Adding a Quit Button

The current CAPS paradigm does not specifically have a quit button. Instead, the user places his cursor in the active prototype window and executes a "Control - C". This section addresses how to make your own quit button. Although it is written in the context of the multiple file approach, the concepts are transferable to the single file approach.

The quit button will, of course, trigger a mouse event which will be detected and acted on by a TAE Event Handler.

Edit the interface as follows. Modify the input panel and select New Item. Create an item with a name that reflects the operation such as "quit_op." The title is "Quit", the presentation category is "Selection", and the Presentation Type is "Push Button" [other options like "Radio Button" will also work]. Click OK when done. Then File/Save.

You are now going to generate code. Refer to instructions elsewhere on how to use the Multi-file Approach.

In pan_mult_in_b.a, under the event handler of quit_op_Event, just add the call "global_b.Set_Application_Done". This procedure sets a global "done" variable from false to true. The code fragment could look like this:

```
if (tae_misc.s_equal(value(1), "Quit"))
Text_IO.put_line("In atacms.pan_gui_in_b.a, user selectedQuit'");
 global.Set_Application_Done; -- sets "done" flag to true
```

Three other packages will periodically test this "done" variable to see if it has changed from false to true.

This is programmed already in the body of file <prototype>.generated_tae_input_event_monitor.a. The call here will shut down the TAE windows. We, however, must make minor changes to the dynamic and static schedulers in file <prototype>.a. Both of these have tasks that run in infinite loops. Make these simple change these to conditional loops with a function call that checks the "done" variable. Add these two lines:

```
with global;
...
```

while not Global.Application_Done loop

Note that this is invisible to CAPS. If you regenerate code on another version, check to see what parts of the above changes survived.

6. Making a new version of a prototype

In this example, let's say you want to create a version 1.2. Change directories to the prototype you want to revise. Make a new directory (mkdir 1.2). Change directory to the new version's directory (cd 1.2). Do a recursive copy of everything that was in the 1.1 directory (cp -rp ../1.1/* .).

INITIAL DISTRIBUTION LIST

1.	Defense Technical Information Center 8725 John J. Kingman Road., Ste 0944 Ft Belvoir, VA 22060-6218	2
2.	Dudley Knox Library Naval Postgraduate School 411 Dyer Road Monterey, CA 93943	2
3.	Director, Training and Education MCCDC, Code C46 1019 Elliot Rd Quantico, Virginia 22134-5027	1
4.	Dr. Ted Lewis, Chairman, Code CS/L Computer Science Department Naval Postgraduate School Monterey, CA 93943	1
5.	Dr. Valdis Berzins, Code CS/Vb Computer Science Department Naval Postgraduate School Monterey, CA 93943	1
6.	Dr. Man-Tak Shing, Code CS/Sh Computer Science Department Naval Postgraduate School Monterey, CA 93943	1
7.	MAJ David Dampier Georgia Tech. 115 O'Keefe Building Atlanta, GA 30332-0862	5
8.	LCDR David S. Angrisani 3327 Watson Rd. St. Louis, MO 63139	2
9.	MAJ George Whitbeck 107 Coopers Ln Stafford, VA 22554	2

10.	Arthur G. Angrisani 2275 Golf Isle Dr. Melbourne, FL 32939	1
11.	Margaret Gates 2266 Sequoia Dr. Clearwater, FL 34623	1
12.	Dr. Luqi, Code CS/Lq Computer Science Department Naval Postgraduate School Monterey, CA 93943	5